

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
13 September 2001 (13.09.2001)

PCT

(10) International Publication Number
WO 01/67707 A2

(51) International Patent Classification⁷: **H04L 29/00**

(21) International Application Number: **PCT/US01/06707**

(22) International Filing Date: 2 March 2001 (02.03.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/186,693	3 March 2000 (03.03.2000)	US
60/186,774	3 March 2000 (03.03.2000)	US
09/695,499	23 October 2000 (23.10.2000)	US
09/753,141	29 December 2000 (29.12.2000)	US

(71) Applicant (for all designated States except US): **SCALE EIGHT, INC. [US/US]; 625 Second Street, Suite 201, San Francisco, CA 94107 (US).**

(72) Inventors; and

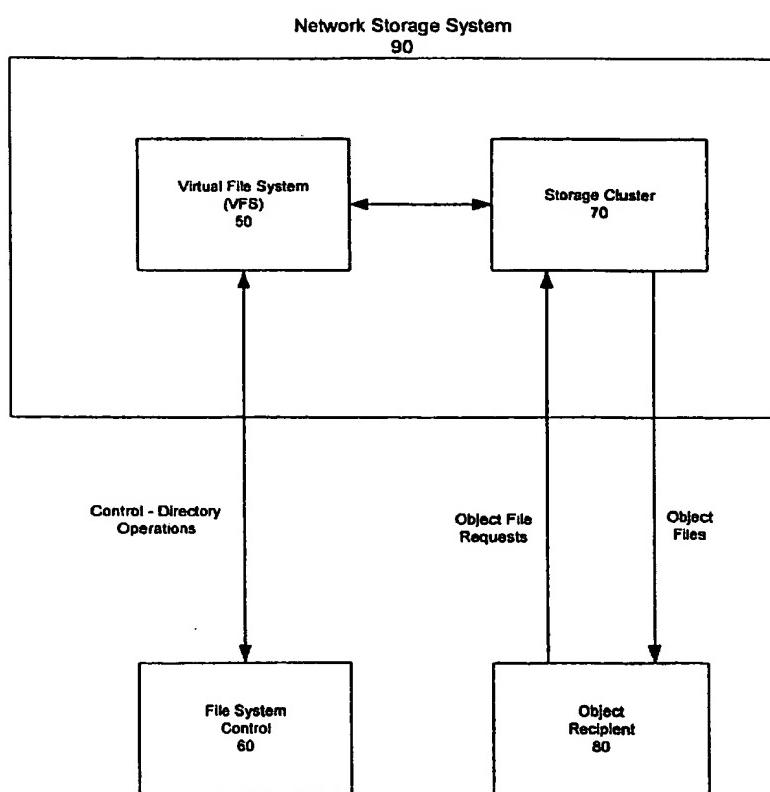
(75) Inventors/Applicants (for US only): **COATES, Joshua, L. [US/US]; 70 Brookwood Road, Orinda, CA 94563 (US). JONES, F., Alan [US/US]; 415 Pope Street, Menlo Park, CA 94025 (US). RUSSEL, Georgina, L. [US/US]; 899 Oak Street #4, San Francisco, CA 94117 (US). GONZALEZ, Michael [US/US]; 20949 Wilbeam Avenue, Castro Valley, CA 94546 (US). BOZEMAN, Patrick, E. [—/US]; 500 Beale Street #311, San Francisco, CA 94105 (US). GAUTIER, Taylor [US/US]; 708 38th Avenue #3, San Francisco, CA 94121 (US). PATTERSON, David, A. [US/US]; 114 Purdue Avenue, Kensington, CA 94708 (US).**

(74) Agent: **STATTLER, John; Stattler Johansen & Adeli LLP, P.O. Box 51860, Palo Alto, CA 94303-0728 (US).**

(81) Designated States (national): **AB, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU,**

[Continued on next page]

(54) Title: A NETWORK STORAGE SYSTEM



(57) Abstract: A network storage system includes a virtual file system ("VFS"), to manage the files of the network storage system, and a storage center that stores the files. The VFS and the storage center are separated, such that a client accesses the VFS to conduct file system operations and the client accesses the storage center to upload/download files. The client accesses the network storage system through one or more storage ports. The storage center includes a plurality of distributed object storage managers (DOSMs) and a storage cluster that includes a plurality of intelligent storage nodes. The network storage system includes additional storage centers at geographically disparate locations. The network storage system uses a multi-cast protocol to maintain file information at the DOSMs regarding files stored in the intelligent storage nodes, including files stored in disparate storage centers.

WO 01/67707 A2

BEST AVAILABLE COPY

A NETWORK STORAGE SYSTEM

BACKGROUND OF THE INVENTION

Field of the Invention:

The present invention is directed toward the field of storage, and more particularly
5 toward accessing remote storage through use of a local device.

Art Background:

With the rapid digitization of music, film and photographs, customer demand is driving the Internet to become the most preferred transport mechanism for all forms of digital media. Using the Internet, users have instantaneous worldwide access to their favorite
10 movies, songs, or personal memorabilia. As the producers and owners of media content increasingly use the Internet as a primary method for worldwide distribution, the aggregate amount of rich media content available over the Internet is increasing at an extremely rapid rate.

Not only is the number of rich media objects available over the Internet growing
15 exponentially, but the size of the media, generally referred to herein as objects, is also dramatically increasing. A median Web object is 5 kilobytes (KB) in size, while the size of a rich media object may be 100 to 1 million times larger. For example, high-resolution digital photographs average 500 KB per picture. Digital music runs 3 to 5 megabytes ("MB") per song, and digital movies may reach up to 4 gigabytes ("GB") in size.

As the number of personal computers, digital camcorders, digital cameras, and personal digital audio players grow, demand for Internet bandwidth to store, share and retrieve media files across the Internet also will grow. As the use of high-bandwidth digital subscriber lines ("DSL"), cable modems, and digital broadcast satellite networks gain in popularity, which supports the growth of the Internet backbone, the demand for using the
20 Internet as a primary delivery channel for rich media objects also gains in popularity. This development causes a virtuous cycle, where the installation of broadband networks drives the use of rich media devices, which in turn, creates demand for further improvements in network bandwidth, and so on.

The distribution of rich media objects across the Internet creates the need for
30 increased storage capacity to store these rich media objects. As the number of personal

Another issue regarding storage of rich media objects is the time to market. The time to market is often a crucial requirement for new rich media Web sites. Growth rates are measured in terabytes per month. Quickly bringing new capacity online becomes a strategic advantage in fast-moving markets. Typically, with traditional storage solutions, it takes a customer two to six months to integrate a fully operational multi-terabytes storage unit with the content providers site. This start-up time is too slow to meet rapidly increasing business demands. Pre-building large amounts of excess capacity in anticipation of this demand is one tactic to deal with unpredictable demand spikes, but this approach is prohibitively expensive.

Traditional storage architectures have been optimized for database and file server applications. The Internet introduces a whole new set of demands on storage devices, including scalability, global access, user accounts, and rapid deployment. With the explosive growth in rich media served over the Internet over the next several years, this is coming to a head. The coming tide wave of rich content will surpass the capabilities of even the most robust enterprise storage architectures. Accordingly, there is a demand to develop new paradigms in new ways of designing Internet ready rich media storage systems.

SUMMARY OF THE INVENTION

A network storage system includes a virtual file system ("VFS") and a storage center. The VFS stores file system information to manage the files of the network storage system. The storage center stores the files of the network storage system. The VFS and the storage center are separated, such that a client accesses the VFS to conduct file system operations and the client accesses the storage center to upload/download files.

In one embodiment, the client accesses the network storage system (*e.g.*, virtual file system and the storage center) through a storage port. The storage port provides access to a client's files of the network storage system. In one embodiment, the client mounts the storage port, through a standard NFS or CIFS operation, and performs file system operations and accesses by issuing local file system operations to the storage port. In response, the storage port translates the local file system operations to network storage system operations. Additional storage ports may be configured at the client to access the network storage system in the event of a failover condition.

In one embodiment, the storage center includes a plurality of distributed object storage managers (DOSMs) and a storage cluster that includes a plurality of intelligent

Figure 11 is a block diagram illustrating one embodiment for implementing a VFS for use with a network storage system.

Figure 12 illustrates example database tables for implementing the file system with a database.

5 Figures 13A and 13B are flow diagrams illustrating one embodiment for performing directory operations in the VFS.

Figure 14 is a flow diagram illustrating one embodiment for the delete file operation for the network storage system.

Figure 15 illustrates geographical replications of storage centers.

10 Figure 16 is a block diagram illustrating one embodiment for replicating the storage centers.

Figure 17 illustrates one embodiment for use of the storage center in a content delivery network.

15 Figure 18 is a flow diagram illustrating one embodiment for use of the storage center with a content delivery network.

Figure 19 illustrates one embodiment for use of the storage port in the network storage system.

Figure 20 is a flow diagram illustrating one embodiment for use of a storage port to deliver content.

20 Figure 21a illustrates one hardware configuration for a storage port device.

Figure 21b illustrates embodiments for implementing the storage port in software.

Figure 22 is a block diagram illustrating one embodiment for a storage port.

Figure 23 is a block diagram illustrating one embodiment for file system translation in the storage port.

25 Figure 24 is a flow diagram illustrating one embodiment for translating a file system operation from a local file system to the network storage file system.

Figure 25 is a block diagram illustrating one embodiment for using the storage port to directly download object files to the end-user.

30 Figure 26 is a flow diagram illustrating one embodiment for directly downloading object files to an end-user.

Figure 27 is a block diagram illustrating one embodiment to interface a storage center to a client's private file directory system.

directory operations. The VFS includes, in part, client assigned filenames and network storage system assigned unique file identifiers for each rich media object. The unique file identifiers are embedded into storage resource locators (“SRLs”).

5 The distributed storage cluster 70 is used to store the object files for the system (*i.e.*, all client data). As shown in Figure 1, the VFS and the storage cluster 70 are coupled to communicate information so as to coordinate file system information with the physical storage of the object files.

10 As shown in Figure 1, file system control 60 issues directory operation requests to the VFS 50. As is described more fully below, file system control 60 may comprise software that uses a library to essentially “translate” file system requests from the client’s local file system to file system requests compatible with the network storage system. In other embodiments, file system control 60 consists of a storage port coupled to the client’s system (*e.g.*, the client’s application or web server). In general, the storage port, implemented in either hardware or software, translates file system commands from the client’s local file system 15 (*e.g.*, NFS or CIFS) to file system requests compatible with the network storage system. In one embodiment, to interface the client’s file system to the network storage system, a client need only mount the storage port as a network drive. The storage port then provides complete access to the network storage system. A detailed discussion of the storage port is set forth below.

20 As shown in Figure 1, object recipient 80 receives, in response to object requests, objects downloaded from storage cluster 70. The object recipient 80 may comprise the client, or the object recipient 80 may consist of one or more end-users. Embodiments for transferring objects from the storage cluster 70 to object recipients, including both end-users and clients, are described more fully below.

25 The network storage system has applications for use as an Internet based media storage service. For this application, the network storage system is an integral part of the Internet infrastructure used by rich media content owners and delivery networks. Figure 2 illustrates one embodiment for use of the network storage system as a media storage service. In general, the storage service 130 provides a single consistent worldwide image of a client’s 30 (*e.g.*, a company operating a web site) entire directory of rich objects. For this embodiment, an end-user 100 is coupled to both the content origin server 120 and storage service 130 through a network. For example, the end-user 100 may be coupled to the content origin server 120 and storage service 130 via the Internet. The storage service 130 includes

wherein "n" is any integer value greater than one. Similarly, there are "n" intelligent storage nodes for the intelligent storage nodes 340 component (*i.e.*, wherein "n" is also any integer value greater than one).

As shown in Figure 3, file upload and download operations are input to a load balancing fabric 310. In one embodiment, the load balancing fabric 310 is a layer four ("L4") switch. In general, L4 switches are capable of effectively prioritizing TCP and UDP traffic. In addition, L4 switches, which incorporate load balancing capabilities, distribute requests for HTTP sessions among a number of resources, such as servers. For this embodiment, the load balancing fabric 310 distributes upload and download requests to one of a plurality of DOSMs based on DOSM availability. The load balancing capability in an L4 switch is currently commercially available.

Each DOSM independently handles hundreds of simultaneous download transactions. In one embodiment described below, each DOSM has a local high-speed disk cache to store frequently accessed file objects. Each DOSM has a map, dynamically generated, of the storage system. The map identifies a correspondence between an intelligent storage node address and an object finger print. In one embodiment, the DOSMs record all usage and performance data gathered by a separate accounting system and monitoring system.

The DOSMs 320 communicate with the intelligent storage nodes 340 via an interconnect fabric 330. The interconnect fabric 330 consists of a high-speed, high bandwidth fabric to ensure that all the DOSMs 320 communicate with every intelligent storage node at all times. In one embodiment, the DOSMs 320 communicate with the intelligent storage node over the interconnect fabric via a protocol, entitled the distributed object storage protocol ("DOSP"). Effectively, the DOSP links hundreds of intelligent storage nodes into one large storage cluster. As described more fully below, the DOSP consist of a multi-cast protocol as well as a point-to-point protocol.

In general, the intelligent storage nodes 340 provide the persistent store for the objects or files. The intelligent storage nodes contain thousands of high-density disk drives. The intelligent storage nodes are described more fully below in conjunction with the discussion of Figure 7.

In one embodiment, the network storage system uses the storage resource locators ("SRLs") to process requests. In one embodiment, the network storage system uses the following format for the SRL:

[http://<storage-cluster>/<encoded-request>/<digital-signature>/<arbitrary-customer-uri,](http://<storage-cluster>/<encoded-request>/<digital-signature>/<arbitrary-customer-uri,>)

Table 3 includes two access method types for the access method field.

Table 3

Access method	Encoding	Comment
8RL	0x0001	End user SRL request.
MediaPort	0x0002	Internal Storage Port request.

Table 4 includes operational codes for the op code field.

Table 4

Operation	Encoding	Arguments
NO_OP	0x0000	None
STORE	0x0010	Pfid – numeric Parent folder id to upload the file to. Other arguments are mime encoded.
FETCH	0x0020	Md5 – alphanumeric Hexadecimal representation of the md5 hash of the file to be downloaded.
FETCH_AUTH	0x0021	Md5 – alphanumeric Hexadecimal representation of the md5 hash of the file to be downloaded. Authentication Callback URI – alphanumeric URL encoded callback URI
DELETE	0x0050	Md5 – alphanumeric Hexadecimal representation of the md5 hash of the file to be deleted.
CONTROL	0x1000	ControlTicket – alphanumeric Hexadecimal representation of the digital signature of the XML control document.

5

Figure 4 is a flow diagram illustrating one embodiment for the download operation in the storage cluster. For purposes of nomenclature, the “recipient” in a download operation is the destination of the file for the download operation. The storage cluster receives a download request, including the unique file identifier (e.g., SRL) (block 400, Figure 4).

- 10 When the storage cluster receives a download request, the load balancing fabric 310 (Figure 3), such as an L4 switch, selects an available DOSM (block 410, Figure 4). The DOSM parses the SRL to extract the certificate and the encoded request (block 415, Figure 4). From the encoded request, a certificate is calculated, and the calculated certificate is compared to the SRL certificate. If the SRL does not authenticate, then an error message is sent to the recipient (blocks 420 and 425, Figure 4). Alternatively, if the SRL does

When the intelligent storage node is located, the DOSM obtains a connection with the intelligent storage node, and opens the file with the requested object. If the storage node is readable (*i.e.*, the DOSM successfully reads the file from the storage node), then the object is transmitted from the intelligent storage node to the recipient via a network (*e.g.*, using HTTP protocol over the Internet). If the object file is not readable, then a failover procedure is executed to obtain the object in a different storage node and/or storage center, and the DOSM obtains a connection with the new storage node (blocks 442, 468 and 435, Figure 4). Thereafter, the object is transmitted from the storage cluster to the recipient (block 495, Figure 4).

In one embodiment, accesses to the network storage system require a valid authentication certificate. In one embodiment utilizing CDNs, the certificate is based on the object file's unique user filename and a secure key assigned to each client account. In other embodiments, the network storage system supports full HTTPS and SSL protocols for secure communications between clients/end-users and the network storage system.

Figure 5 is a flowchart illustrating one embodiment for authentication in the network storage system. To authenticate a request, the network storage system decodes the SRL to extract the client identification, the SRL certificate and the client filename or object fingerprint (block 500, Figure 5). The network storage system (*i.e.*, virtual file system or storage cluster) extracts a "secret" or secure key corresponding to the client identified with the request. In general, the "secret" or secure key is a password supplied by the client to authenticate operations in the network storage system. Using the secure key and object fingerprint, the network storage system generates a calculated certificate (block 520, Figure 5). In one embodiment, the network storage system generates a calculated certificate for the request in accordance with the following expression:

$$\text{MD5 Hash}(\text{Secure Key} + \text{MD5 Hash}(\text{Secure Key} + \text{Encoded SRL}))$$

As shown above, a first MD5 hash calculation is performed on the object fingerprint and the secure key to obtain a first result, and a second MD5 hash calculation is performed on the first result and the secure key to obtain the calculated certificate. The network storage system compares the calculated certificate with the SRL certificate (*i.e.*, the certificate transmitted with the SRL request) (block 530, Figure 5). If the certificates match, then the SRL is authenticated, and the request is performed (blocks 540 and 560, Figure 5). Alternatively, if the calculated certificate does not match the SRL certificate, then the

intelligent storage nodes are healthy, how much space is on the disks, etc. In one embodiment, as shown in Figure 6, state table 630 stores: read-write state of the storage nodes; health of the storage nodes (including an identification of failed nodes); and the current load of the storage nodes, including available storage capacity and number of input/output ("I/O") operations per second. The DOSM uses state information to select, in an upload operation, the appropriate intelligent storage node for storage of a new object file. For example, the DOSM uses information on the number of input/output ("I/O") operations per second to load balance the storage nodes. The DOSM also uses information on available storage capacity to select an intelligent storage node to store a new object file.

Figure 7 is a block diagram illustrating one embodiment for an intelligent storage node. For this embodiment, the intelligent storage node is implemented on a computer, including software to perform the functions described herein. An intelligent storage node 700 includes a processing core 710 that consists of one or more central processing units ("CPUs"). In one embodiment, the processing core 710 comprises two CPUs. The intelligent storage node 700 also includes volatile memory, labeled 730 in Figure 7. The memory 730 is used to store instructions executed by the processing core 710, as well as data used by the intelligent storage node. The intelligent storage node 700 further includes a network interface 720 to interface the intelligent storage node to the plurality of distributed object storage managers 320 via the interconnect fabric 330. The elements of the intelligent storage node 700 communicate via a computer transport mechanism 750 (e.g., a peripheral component interconnect ("PCI") bus, processor bus, etc.). The computer transport mechanism 750 is intended to represent a broad category of one or more computer busses, such as peripheral component interconnect ("PCI") bus or the industry standard association ("ISA") bus.

The intelligent storage node 700 further includes a plurality of disk drives 740 to store the object files. As shown in Figure 7, the number of disks in an intelligent storage node is represented as "n", such that "n" is an integer value greater than one. In one embodiment, the processing core 710 communicates with the disk drives 740 using the ISA protocol. However, any protocol used to access the disk drive, including standard computer serial interface ("SCSI") protocol, may be used without deviating from the spirit or scope of the invention.

The intelligent storage node contains information to identify object files that it stores. In one embodiment, the information to identify object files is stored in the file system

files ensures accessibility to the object in the event a failure occurs in an intelligent storage node. In one embodiment for “mirroring” the object files, the network storage system stores the object file at different geographic locations (*e.g.*, different storage centers). If access to the geographically disparate storage center is unavailable at the time the object file is uploaded, then an additional copy of the file is stored at the local storage center.

In one embodiment, the DOSM uses a state table (Figure 6) to select the intelligent storage nodes most appropriate to store the new object. For purposes of discussion, the selected intelligent storage nodes are referred to herein as the “destination intelligent storage nodes.” The DOSM establishes a connection with the destination intelligent storage node (block 850, Figure 8). In one embodiment, the DOSM establishes a DOSP point-to-point connection with the destination source node. The object file is then transferred to the destination intelligent storage node (block 860, Figure 8). In addition, after transferring the file to the intelligent storage node, the DOSM receives a status message as part of the DOSP point-to-point protocol. The status message indicates whether the transfer operation was successful.

In one embodiment, the destination intelligent storage node generates a unique fingerprint for the object file (block 870, Figure 8). Specifically, the destination intelligent storage node computes an MD5 hash of the contents of the object file. The intelligent storage node also verifies the object file. After receiving the successful status at the DOSM, the DOSM establishes a connection to the virtual file system (“VFS”). The DOSM communicates file information (*e.g.*, the 128 bit MD5 unique object fingerprint, file size, etc.), directory information (*e.g.*, folder ID, parent folder ID, etc.), as well as client information and metadata (block 880, Figure 8). The VFS attempts to verify the upload. If the VFS does not verify the upload, then an error message is sent to the source of the upload request (blocks 890 and 835, Figure 8). If the VFS does verify the upload, then the verification is transmitted to the DOSM. In turn, the DOSM verifies the upload to the source (block 895, Figure 8). Also, the storage system returns, to the source, a file handler that uniquely identifies the file to the network storage system.

If the source of the upload request is an end-user, then the DOSM re-directs the end-user to the client. For example, the DOM may redirect the end-user to a predetermined URL at the client’s web site. In other embodiments, if the source was a storage port, then the DOSM transmits a storage system node (*i.e.*, handler used only for the storage system) and the unique object file fingerprint.

(“XML”). Although the VFS is described using the HTTP protocol with XML encoded requests, any network protocol with any type of request format may be used without deviating from the spirit or scope of the invention.

In one embodiment, the VFS employs a database to implement the file system. For 5 the database implementation, each directory operations request is converted into the database operation. Alternatively, the VFS may implement the file system using a local file system (*i.e.*, a file system local to the VFS). For the file system embodiment, files are generated to store information stored in the database implementation. Also, the DDMs include a lookup table to locate the files in the distributed directories. The files or database tables are 10 replicated in a remote storage center.

The network storage file system consists of files arranged in directories or folders (hereafter referred to as folders). Similar to most file systems, the network storage file system is a hierarchical file system. In a hierarchical file system, directories or folders are arranged in levels, starting with a root or base folder. Additional folders or sub-folders are 15 then arranged under the root folder. The file system may comprise any number of levels, such that additional layers of sub-folders fall beneath other sub-folders. For purposes of nomenclature used herein, a parent folder to a folder is the folder arranged above the folder in the hierarchy of folders or directories.

Figure 12 illustrates example database tables for implementing the file system with a 20 database. For the database embodiment, the VFS maintains a customer table 1200, folder table 1210 and file table 1220. The customer table 1200 includes fields for “customer ID”, “customer name”, and “customer reserved fields.” The customer ID is a network storage system identifier used to uniquely identify the client. The customer name is the real name associated with a customer. For the first example entry in the customer table 1200, 25 “customer A” has a customer ID of “1.” The customer reserved fields provide storage reserved for use by the client.

The folder table 1210 includes fields for “customer ID”, “folder ID”, “folder parent ID”, and “metadata.” For this embodiment, each entry in the folder table corresponds to a folder in the network storage file system. The customer ID, the same customer ID stored in 30 the customer table, uniquely identifies the client. For the example entries in folder table 1210, the customer ID of “3” identifies that the folders have been assigned to “customer C.” The folder ID identifies the specific folder for that entry. For example, the first entry in folder table 1210 is for a folder identified by the identification of “2.” The third column,

Figure 12, if the “open folder” request included the arguments “folder ID = 2” and “customer ID = 3”, then the DDM extracts, from the folder table in the distributed directory, folder IDs 100 and 251 (*i.e.*, folders 100 and 251 are sub folders of the root folder 2). If the “open folder” request included the arguments “folder ID = 100”, then the DDM extracts from the 5 file table file handlers “52.MD5” and “55.MD5.”

If the operational code in a directory request is for an “open file” operation, subsequent to an “open folder” request, then file information is obtained from the file table (*i.e.*, file handler) and the client table (*i.e.*, client identification) to construct an authentication certificate and an SRL for the file. For the above example, if the argument with the “open 10 file” operation specified the file “52.MD5”, then file and client information are obtained to construct the SRL for the “52.MD5” file.

If the operational code in a directory request is for a “move folder” operation, then a database operation is performed to revise the entries in the file and folder tables to reflect the new location of the folder. The “move folder” operation includes, as an argument, the new 15 destination for the folder. Using the example of Figure 12, if the “move folder” operation specified moving folder ID 166 from a sub folder of folder ID 251 to directly beneath the root folder 2, then the parent folder ID on the fourth entry of folder table 1210 is modified from “251” to “2.” Also, for file table 1220, the parent folder ID for the third and fourth entries are modified from “251” to “2.”

20 If the directory operation is a “create folder” operation, then a new entry or row is generated for the folder table (blocks 1360 and 1365, Figure 13A). The “create folder” operation includes a parent folder as an argument. As described below, the client’s folder name is converted to the network storage systems folder identification. Using the example of Figure 12, if the requester desires to create a new folder under the sub folder 166, then the 25 DDM assigns a new folder identification for the folder, and enters a new row or entry for the folder table 1210 with a folder parent ID of 166.

If the directory operation is a “move file” operation, then a database operation is performed to revise an entry in the file table to reflect the new location of the file (blocks 1370 and 1375, Figure 13A). The “move file” operation includes a new destination for the 30 file as an argument in the directory request. For the example database tables in Figure 12, if the “move file” operation specified moving file “52.MD5” from folder 100 to folder 166, then the folder ID and folder parent ID fields for the first entry of file table 1220 are revised to “166” and “251”, respectively.

validation check (blocks 1400 and 1405, Figure 14). If the delete request is not valid, then an error message is transmitted to the requester (blocks 1410 and 1415, Figure 14). If the request is validated, then the DDM extracts a file handler (*i.e.*, MD5 file handler) from the file table in the database (block 1420, Figure 14). The DDM deletes the file identification 5 from the file table in the database (block 1450, Figure 14). In addition, the DDM constructs a delete SRL, and transmits the delete SRL to the storage cluster (block 1460, Figure 14). In response to the delete operation, the storage cluster extracts the reference count for the corresponding file. If the reference count is greater than 1, the storage cluster decrements the reference count by one (blocks 1430 and 1440, Figure 14). Alternatively, if the reference 10 count is one, the storage cluster decrements the reference count to zero, and deletes the file identified by the SRL, in the appropriate intelligent storage node (block 1470, Figure 14).

Dynamic Data Caching:

Figure 10 is a block diagram illustrating one embodiment for caching data in the storage cluster. As shown in Figure 10, there are “n” DOSMs. Each DOSM (*i.e.*, DOSM 1 15 DOSM 2, DOSM 3 … DOSM “n”) contains a corresponding data cache (*i.e.*, data cache 1 data cache 2, data cache 3 … data cache “n”). The network storage system file upload and download operations are received by the load balancing fabric 310 (also see Figure 3). A switch, such as an L4 switch, with load balancing capabilities, allocates resources among a pool of resources. For the network storage system, the load balancing fabric 310 efficiently 20 allocates requests among the “n” DOSMs. In one embodiment, when a DOSM transfers an object from the intelligent storage node to a destination, the object is cached in the data cache of the corresponding DOSM. Objects are deleted from the data cache in order to store objects more recently requested via a least recently used (“LRU”) caching policy.

Load balancing the DOSMs in the network storage system permits an “automatic” 25 caching of objects in high demand. In prior art systems, elaborate mechanisms are employed to identify data in high demand. Based on these decision mechanisms, data is cached in an attempt to meet the needs of the high demand. For example, an object may be in high demand when a movie studio offers, over its web site, a video preview of a newly released or upcoming film. For this example, the movie studio uses the network storage system to 30 deliver the media rich object, “New Film Preview.” The “New Film Preview” may be available to the end-user if the end-user “clicks” on a URL in the movie studio’s web site. For this example, if the movie is very popular, when the movie studio client offers the “New

view of their private file system. Thus, the network storage system supports simultaneous downloads of a single object that appears identical to users worldwide. In one implementation, the network storage system spans multiple continents with storage repositories or storage centers. The automatic geographic load balancing between storage centers ensures that all requests are directed to the nearest storage center. However, to provide fail over and enhanced performance, the storage center, including the storage cluster and VFS, are replicated. The physical replication across multiple locations includes a traffic management service. The traffic management service provides geographic load balancing of user transactions among geographic locations.

Figure 15 illustrates geographical replications of storage centers. For this example, there is a North American storage center 1510, an Asian storage center 1530, and a European storage center 1520. As shown in the example of Figure 15, clients and end-users in North America have optimal access to the storage center through the North American storage center 1510. Also, clients and end users in Europe have optimal access to European storage center 1520. Similarly, clients and end-users in Asia have optimal access to be Asian storage center 1530. In this configuration, the storage center is coupled to a wide area network to provide the maximum bandwidth for the delivery of objects. If a particular storage center becomes overloaded with requests, new requests are automatically diverted to the next closest storage center. All objects are geographically mirrored to provide one hundred percent disaster protection. Also, if access to the geographically disparate storage center is unavailable at the time a file is stored, then an additional copy of the file is stored at the local storage center (*i.e.*, the object file is mirrored locally).

The components within the network storage system are fully redundant with automatic recovery. Thus, the system supports extremely high level of service availability.

Download requests to each geographic storage center are continuously distributed across the DOSMs to deliver the fastest possible response time. In addition, in one embodiment, a global load balancing system ensures that the worldwide load across all storage centers is evenly spread to eliminate any “hot spots” and alleviate transitory demand spikes. The storage system operates far more quickly than the network itself, and thus introduces negligible delay to the overall file transit time. Thus, the worse case elapsed time for the individual object download is primarily determined by the speed of the wide area network used to transfer the object.

object storage managers of storage center 1520 are coupled to the interconnect fabric of storage center 1510. Based on this configuration, the distributed objects storage managers of storage center 1510 have access to the intelligent storage nodes of storage center 1520. Likewise, the distributed object storage managers of storage center 1520 have access to the intelligent storage nodes of storage center 1510. As discussed above, each DOSM maintains a lookup table that correlates a file to an IP address (See Figure 6). For example, if a file specified in a download request resides on storage node 1 in storage center 1510, then an entry of the DOSM lookup table specifies IP addr₁. Similarly, in storage center 1520, if a file resides in storage node 1, an entry for the DOSM lookup table specifies IP addr₁'.

The storage center architecture supports a "dynamic" fail over. If a storage node, or a disk drive on a storage node, renders the access to a file inaccessible, then the DOSM may obtain the file from the replicated storage center. In one embodiment, to perform "dynamic" fail over, a mapping is stored between intelligent storage nodes in storage center 1510 and intelligent storage nodes in storage center 1520. Table 6 below shows a mapping for the example in configuration of Figure 16.

Table 6

IP Address	IP Address'
IP Addr ₁	IP Addr ₁ '
IP Addr ₂	IP Addr ₂ '
...	...
IP Addr _n	IP Addr _n '

For this example, IP addr₁ maps to IP addr₁'. If there is a failure in storage node 1 in storage center 1510, then DOSMs of storage center 1510 access storage node 1 of storage center 1520 using IP addr₁'. In one embodiment, the IP mapping between storage centers is implemented by modifying only the subnet address portion between the two IP addresses mapped. For example, if IP addr₁ is 10.3.100.1, then IP addr₁' is derived by changing, as appropriate, the subnet portion of the address (e.g., 10.10.100.1).

The directory information stored in the VFS is replicated between storage center 1510 and 1520 in a similar manner. Thus, if a failure occurs in a distributed directory of storage center 1510, then the distributed directory manager in storage center 1510, using an IP address mapping, accesses the replicated distributed directory in storage center 1520.

For purposes of illustration, a wide area network 1750 is shown as including satellite communication networks 1760, wireless communication networks 1770, and fiber-optic networks 1780. As illustrated in Figure 17, the CDN server 1730 is located close to the edges of the wide area network 1750. The location of CDN server 1730 close to the wide area network 1750 optimizes the delivery of objects cached at the CDN server 1730. For this embodiment, one or more storage center(s) 1710 are coupled to the CDN server 1730. In the event of a cache miss at the CDN server 1730, the CDN server 1730 obtains the content (*e.g.*, object file) from storage center(s) 1710. This configuration allows the CDN server 1730 to bypass the slower content origin web server 1720 in the event that content, requested by end-user computer 1740, is not located at the CDN server 1730. According, the storage center(s) 1710 optimize routing of content through the Internet back to the CDN when the desired content is not located in the local cache.

Figure 18 is a flow diagram illustrating one embodiment for use of the storage center with a content delivery network. The end-user, through the end-user computer, generates an HTTP request to the content origin web server (block 1800, Figure 18). In response to the user request, the content origin server returns to the end-user computer HTML with embedded file URLs (block 1810, Figure 18). The embedded file URLs identify the rich media objects stored at the CDN server. To obtain the rich media objects, the end-user computer generates HTTP file requests to the content delivery network (*e.g.*, CDN server 1730) (block 1820, Figure 18). If the file identified by the URL is located in a cache at the CDN server site, then the CDN server delivers the file to the end-user computer (blocks 1825 and 1850, Figure 18). Alternatively, if the file is not cached at the CDN server site, the CDN server generates an HTTP file request to the storage center (blocks 1825 and 1830, Figure 18). In one embodiment, the HTTP file request includes the network storage system's SRL, to uniquely identify the file. In response to the CDN server's request, the storage center downloads the file to the CDN cache (block 1840, Figure 18). The CDN server delivers the file to the end-user computer (block 1850, Figure 18).

Accessing The Network Storage System:

There are multiple ways to access the network storage system. In one embodiment, the client uses a "storage port." The storage port provides access to the network storage system through a standard file system interface (*e.g.*, network file system ("NFS") or Microsoft NT CIFS). The storage port may be configured by the client in various ways for

storage port 1930 with the content web server 1925, the storage port 1930 is mounted as a storage device. In one embodiment, one directory is mounted for object files and a second directory is mounted for SRLs. As shown in Figure 19, the storage port 1930 communicates with the storage center 1950 to conduct file and directory operations.

5 Figure 20 is a flow diagram illustrating one embodiment for use of a storage port to deliver content. The client site receives a URL file request from an end-user computer (block 2010, Figure 20). The URL identifies an object file associated with the client's web site. In response to the end user's URL file request, the client site (*e.g.*, content web server) generates a local file system request for the object file (block 2020, Figure 20). The local file system request is received by the storage port. The storage port includes a cache to store both object files and directory information. If the object file is stored locally in the storage port, then the storage port retrieves the object file from the data cache, and returns the object file to the content web server in response to the local file system request (blocks 2030, 2040, and 2070, Figure 20). Alternatively, if the storage port does not store a copy of the object file in its data 10 cache, then the storage port requests the object file from the storage center (blocks 2030 and 2050, Figure 20). In response to the local file system request, the storage center downloads the object file to the storage port, and the object file is returned to the content web server (blocks 2060 and 2070, Figure 20). Thereafter, the content web server delivers the object file to the end-user in response to the URL file request (block 2080, Figure 20).

20 The storage port may be implemented in either hardware or software. Figure 21a illustrates one hardware configuration for a storage port device. As shown in Figure 21a, the content web server 2100 communicates with the storage port 2110 over a communications link 2120, such as a local area network. The storage port 2110 conducts file and directory operations with storage center 2130.

25 Figure 21b illustrates embodiments for implementing the storage port in software. In one embodiment, the network storage system is accessed through library calls or through application program interface ("API") calls. For these embodiments, the software provides translation between the client's local file system and the network storage file system. As discussed above, the storage center 2160 includes software running on computers for 30 performing the functions of the VFS and intelligent storage clusters. This software includes entry points (*i.e.*, APIs) to permit interfacing of external software. In part, the APIs on the storage center software permit the client to conduct file and directory operations as described herein. As shown in Figure 21b, content web server 2140 runs, in addition to software to

translator 2320 and storage system access processes 2330. The file system translator 2320 includes local file system interception 2340 and storage system kernel processes 2350.

In operation, local client file system 2310, which may include operating system software running at the client's site, issues local file system operations. For example, the 5 client software may issue requests, in accordance with UNIX or Microsoft NT to open a file. The file open operation includes a file descriptor that identifies the file in the local file system. Typically, file system calls are processed by the operating system kernel (labeled 2360 in Figure 23). The operating system kernel software maintains a mapping between file descriptors and directories to "inodes." The inodes provide the system a physical pointer to 10 the file data in the system (e.g., a pointer to the file stored on a hard disk drive).

For the embodiment of Figure 23, when the local client file system 2310 issues a file system operation, local file system interception 2340 "traps" or intercepts the call, and passes the thread of execution to the storage system kernel processes 2350. In one embodiment, the 15 local file system interception 2340 comprises CODA software, developed at Carnegie Mellon University. In general, CODA is a type of distributed file system. A portion of the functionality provided by the CODA software exports an underlying file system. Specifically, CODA exports file system operations, typically executed in the kernel level, to applications programs accessible in the user portion of memory. Although file system translation is described using CODA to intercept local file system operations, any software that intercepts 20 file system calls may be used without deviating to the spirit or scope of the invention.

In general, the storage system kernel processes 2350 obtains network storage system file handlers (referred to herein as "storage handlers") for storage in operating system kernel 2360 to provide a mapping between local file system descriptors and storage handlers. Thus, the file descriptors provide a handle to identify files and directories in the local file system, 25 and the storage handlers provide a handle to identify files and directories in the network storage system.

To maintain the mapping between local file system descriptors and storage handlers, the storage system kernel processes 2350 obtains network storage file system information from storage system access processes 2330. Specifically, storage system kernel processes 30 2350 obtains from storage system access processes 2330 storage handlers and directory information. As shown in Figure 23, storage system access processes 2330 obtain directory and storage handler information from directory cache 2370. Alternatively, if directory and storage handler information is not cached at the storage port, storage system access processes

files and subfolders within the subject folders (*i.e.*, the folder that is the subject of the “Open Folder” request). In one embodiment, in response to a request for folder information, the VFS returns name, folder identification, client metadata, upload SRL, and parent folder identification. In response to a request for file information, the VFS returns name, file identification, client metadata, download SRL, and parent folder identification. In one embodiment, the client metadata fields are used to track and maintain state information used in the local file system (*e.g.*, information for UNIX, Microsoft Windows or NT, etc.). In addition to obtaining additional directory information, if the client local file system command is a directory operation (*i.e.*, “move folder”, “delete folder”, etc.), then an XML request to the VFS is generated to perform the directory operation in the VFS. The directory information is received and stored in the directory cache (block 2480, Figure 24).

If the file system operation requires file data (*e.g.*, open file, read file etc.), then the storage port determines whether the file is located in the data cache (block 2440, Figure 12). If the file is stored in the data cache, then the file, or appropriate portion, is transferred from the storage port to the client requestor (block 2090, Figure 12). Alternatively, if the file is not in the data cache, then the storage port generates a file download request to the storage cluster (block 2050, Figure 24). In response to the storage cluster request, the storage port receives and subsequently caches the object file in the data cache (block 2060, Figure 12). The object is then transferred from the storage port to the client requestor (block 2090, Figure 12).

20 2. End User Network Storage System Access Method:

In another embodiment, the storage port supports file downloads directly to the end-user or through a CDN partner. In one embodiment, the SRLs are directly embedded into the Web page HTML, and are sent to the end-user. This results in transferring objects directly from the storage center to the end-user browser. Figure 25 is a block diagram illustrating one embodiment for using the storage port to directly download object files to the end-user. For this configuration, an end-user computer 2610 communicates with a client site 2620 and the storage center 2650. The client site 2620 maintains a web site. For this embodiment, the client site 2620 maintains a web site through a content web server 2630. However, any configuration of servers, including remote web site hosting, may be used without deviating the spirit or scope of the invention.

The content web server 2630 communicates with the storage port 2640, and in turn, the storage port 2640 communicates with the storage center 2650. As illustrated in Figure 25,

file contains an SRL to identify an object file of the network storage system. In one embodiment, to embed the SRL into the web page HTML, the client reads the contents of the shadow file for the corresponding object file. In one embodiment, the shadow file is generated during an upload operation. The client may access a shadow file by mounting the 5 second directory. For example, a client may specify, for the file “foo.text”, the following directory-filename:

storagefilesystem:/export/dir/foo.text.

The client uses this directory and filename to access the contents of the object file, “foo.text.” To obtain the SRL for the example file “foo.text”, a client mounts a different directory, such 10 as the following example directory:

storagefilesystem:/SRL/dir/foo.text,

wherein, the SRL file contains a unique file identifier and the SRL authentication certificate for the file, “foo.text.” To deliver the SRL to the end-user, the client reads the contents of a shadow file for the corresponding object file, and publishes the SRL to the user.

15 3. Client Private File System Directory:

The network storage system of the present invention also supports using an existing private file directory to access the storage system. For this embodiment, the network storage system customer (*e.g.*, client) may desire to use their own file structure in conjunction with the network storage system’s file system. In other embodiments, a client of the network 20 storage system may wish to develop a file system to track additional information beyond that information tracked using NFS or CIFS.

Figure 27 is a block diagram illustrating one embodiment to interface a storage center to a client’s private file directory system. In one embodiment, the storage port at the client site 2820 is replaced with a private file manager 2840. For this embodiment, the private file 25 manager 2840 generates SRLs for object files using a unique file identification assigned to the user file at the time of upload, as well as using a shared secret to authenticate file system operations. As shown in Figure 27, the content web server 2830, operating at the client site 2820, generates file system requests to the private file manager 2840. In turn, the private file manager 2840 issues SRLs corresponding to the object files that are the subject of the 30 request. In one embodiment, the client supplies their own unique ID at the time the client

file identifiers, and allows the customer to interface with the network storage system by only designating unique file names.

Failover Architecture:

In one embodiment, the storage port supports failover or failsafe architectures.

- 5 Figure 29 is a block diagram illustrating one embodiment for a storage port fail over configuration. For purposes of explanation, Figure 29 illustrates a fail over configuration with two storage ports. However, the storage port fail over configuration may be extended to any "2N" fail over configuration. For this embodiment, the fail over configuration includes an active storage port 3010 and a passive storage port 3020. Each storage port includes a
10 plurality of network interface cards. Both the active storage port 3010 and passive storage port 3020 communicate to storage center(s) over wide area network 3065, through network interface cards 3045 and 3025, respectively. The active storage port 3010 and passive storage port 3020 also communicate to the client site network via network interface cards 3050 and 3035, respectively. As shown in Figure 29, the client accesses the active storage port 3010
15 over client site network 3060 using IP Addr.

- For the embodiment of Figure 29, a third network interface card is contained on both the active storage port 3010 (3055) and passive storage port 3020 (3030) to communicate between the devices for fail over monitoring. The active storage port 3010 operates as current storage port at the client site. The passive storage port 3020 monitors the health of
20 the active storage port 3010. Specifically, active storage port 3010 includes health monitoring 3070 that continually executes a process to ascertain the health of the active storage port 3020 (*e.g.*, health of the CPUs, hard disk drives, etc.). For this embodiment, the passive storage port 3020 queries the active storage port 3010 for health status. If a condition occurs in the active storage port 3010 that warrants a fail over condition, then the passive
25 storage port 3020 becomes the active storage port (*i.e.*, the passive storage port is used to interface the client site to storage center(s)).

- In one embodiment, to support fail over, one IP address is used for the NFS/CIFS export. For this embodiment, a standard IP switch over scheme may be utilized. Specifically, when a fail over condition occurs, the passive storage port 3020 assumes the IP
30 address of the active storage port 3010. The health monitoring 3070 and 3080 include both active and passive processes, so that if a fail over condition occurs, the passive storage port may execute the active storage port process.

DOSM may issue the multicast protocol request to local storage nodes (*i.e.*, storage nodes located at its storage center).

Each storage node that receives the multicast request determines whether it contains the requested object file (block 3240, Figure 31). If none of the storage nodes contain the object file, then the DOSM may issue another multicast protocol request at a remote storage location (blocks 3245 and 3247, Figure 31). Again, at the remote storage center, each storage node determines whether it contains the requested object file (block 3240, Figure 31). In another embodiment, if the DOSM does not locate the file using the multicast protocol, the DOSM may query each individual storage node using the DOSP point-to-point protocol.

When a storage node locates the requested object file, the storage node broadcasts the file identification information using the multicast protocol (block 3250, Figure 31). Each DOSM snoops, using the multicast protocol, to receive the file identification information (block 3260, Figure 31). As illustrated in the process embodiment of Figure 31, the multicast protocol may be used to synchronize file location information in the DOSMs in the event of a fail over condition.

Multi-Cast Protocol:

The multi-cast protocol of the present invention supports the maintenance of file information in a distributed storage system. Since the network storage system consists of a plurality of storage nodes, the multicast protocol is used to track file information and synchronize file information throughout the network storage system. The tracking and maintaining of file and directory information includes maintaining information throughout geographically disparate storage centers. In one embodiment, the multi-cast protocol synchronizes cache information in the DOSMs. For example, if a new object file is loaded, the multi-cast protocol provides a means for all DOSMs in the network storage system to obtain information necessary to access the new object file. In addition, some file operations, including delete file or update file operations, require updating the DOSM lookup tables. Also, if a storage node fails, and a fail over condition is executed, the multi-cast protocol provides a means for the DOSMs to locate the file at the storage node the file has been replicated.

To facilitate gathering of information about the system, the DOSP provides several multicast-based services. In one embodiment, these services work in a manner very similar to the non-multicast aspect of the protocol. Specifically, requests consist of three parts: an opcode; a request In structure; and any additional data.

5 Responses consist of a response structure containing a RETURN value and any other return values required to satisfy the request. If data is streamed, a size field precedes the data, followed by the data, and then followed by the Out structure.

10 Since multicast traffic occurs on a completely separate port from point-to-point dosm/dosd traffic, the multicast In/Out structures are not multicast-specific. This makes it possible for the DOSM to query the entire dosd storage cluster or to query an individual machine with the same request/response structures and their associated operational sequencing.

15 One of the jobs of the DOSM is to monitor the current state of nodes in the cluster. There are several tools to facilitate this task. Primarily, the various dos daemons multicast heartbeats on a specific multicast port and group. The DOSM contains an option to query a specific disk, or all of the disks on a given a storage node. A "get disk state" function returns a value, and an array of disk state values (online, offline, down) with one entry per disk. A "get disk status" function contains an option to query a specific disk, or all of the disks on a given a node. The "get disk status" contains a RETURN value, and an array of disk 20 statistics; one array per statistic (bytes free, bytes available, inodes used, inodes available, number of outstanding ops), with one entry per disk. The DOSP includes a load balancing function.

The DOSP includes a heartbeat function. This allows querying specific machines for a heartbeat in addition to providing system-wide tracking functionality via multicast methods.

25 Although the present invention has been described in terms of specific exemplary embodiments, it will be appreciated that various modifications and alterations might be made by those skilled in the art without departing from the spirit and scope of the invention.

5. The method as set forth in claim 1, wherein:

the step of transmitting content comprises the step of transmitting hyper-text mark-up language ("HTML") content; and

5 said SRL into said HTML.

6. The method as set forth in claim 4, wherein the step of embedding said SRL into said content comprises the steps of:

storing at least one SRL for a file in an SRL file; and

10 extracting said SRL from said SRL file.

7. The method as set forth in claim 4, wherein the step of embedding said SRL into said content comprises the steps of:

coupling a local device comprising a cache to said content server;

15 storing at least one SRL for at least one file in said cache of said local device; and
extracting said SRL from said cache of said local device.

8. The method as set forth in claim 7, further comprising the step of: mounting said local device as a storage device for said content server for access to said SRLs.

20

9. The method as set forth in claim 1, further comprising the steps of:
storing at least one SRL for at least one file in an SRL file;
storing said file for access by a file system; and
organizing said SRL files in a file system, accessible to said content server, with a file
25 structure substantially similar to said file structure for said files.

10. The method as set forth in claim 1, wherein:

the step of transmitting a request for said file from said end-user computer to a remote storage center comprises the step of transmitting a hyper-text transfer protocol ("HTTP")
30 request; and

the step of transmitting said file from said storage center to said end-user computer comprises the step of transmitting said files using HTTP.

17. The system as set forth in claim 15, wherein:
said SRL further comprising an SRL file; and
said content server further comprising processes for extracting said SRL from said
SRL file.

5

18. The system as set forth in claim 12, further comprising a local device, coupled
to said content server, that includes a cache for storing at least one SRL for at least one file in
said cache of said local device, wherein said content server further comprising processes for
extracting said SRL from said cache of said local device.

10

19. The system as set forth in claim 18, wherein said content server comprises
processes for mounting said local device as a storage device to said SRLs.

20. The system as set forth in claim 19, further comprising a file system,
15 accessible to said content server, including at least one SRL file for storing at least one SRL,
wherein said file system comprising a file structure substantially similar to a file structure for
said files.

21. A storage center comprising:
20 storage for storing a plurality of files;
storage control for receiving a request from an end-user computer, remote from said
storage center, for at least one file, and for transmitting said file to said end-user computer,
said request comprising at least one storage resource locator ("SRL") corresponding to said
file, and wherein said SRL comprises a unique file identifier to identify said file associated
25 with content that said end-user computer downloaded from a content server.

22. The storage center as set forth in claim 21, wherein:
said SRL further comprises an authentication certificate; and
said storage center for determining whether said request is valid using said
30 authentication certificate, and for transmitting to said end-user computer said file only if said
request is valid.

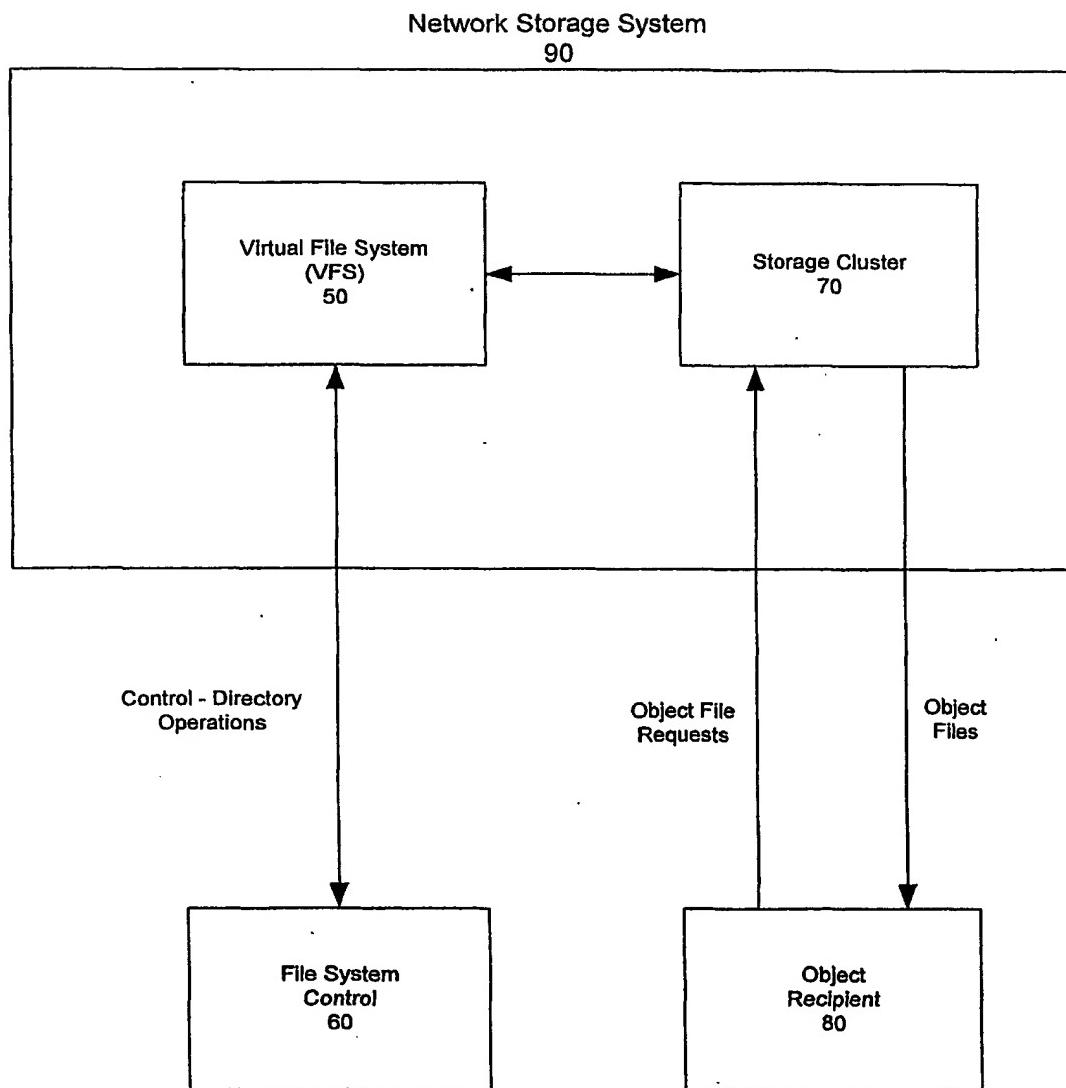


Figure 1

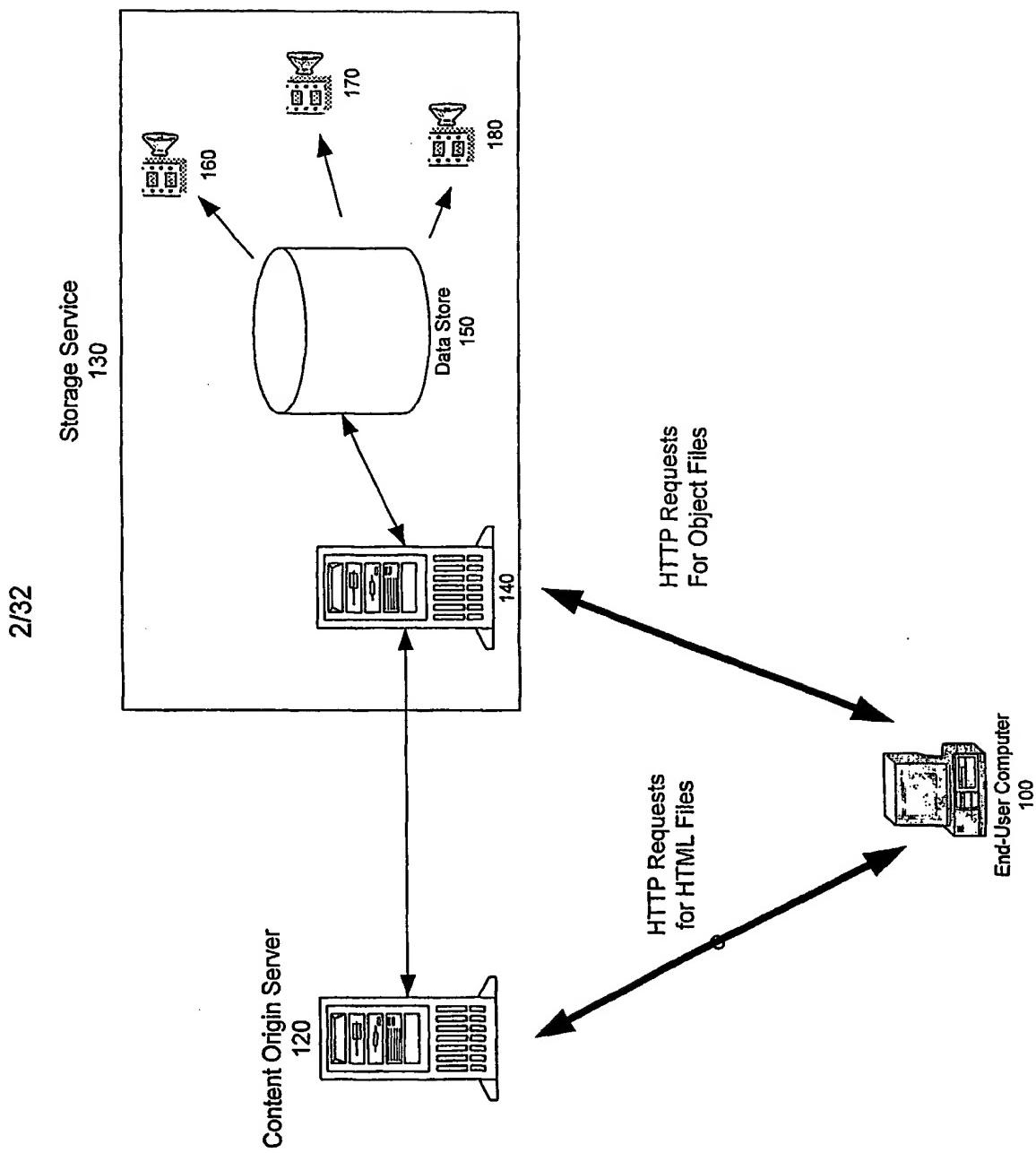
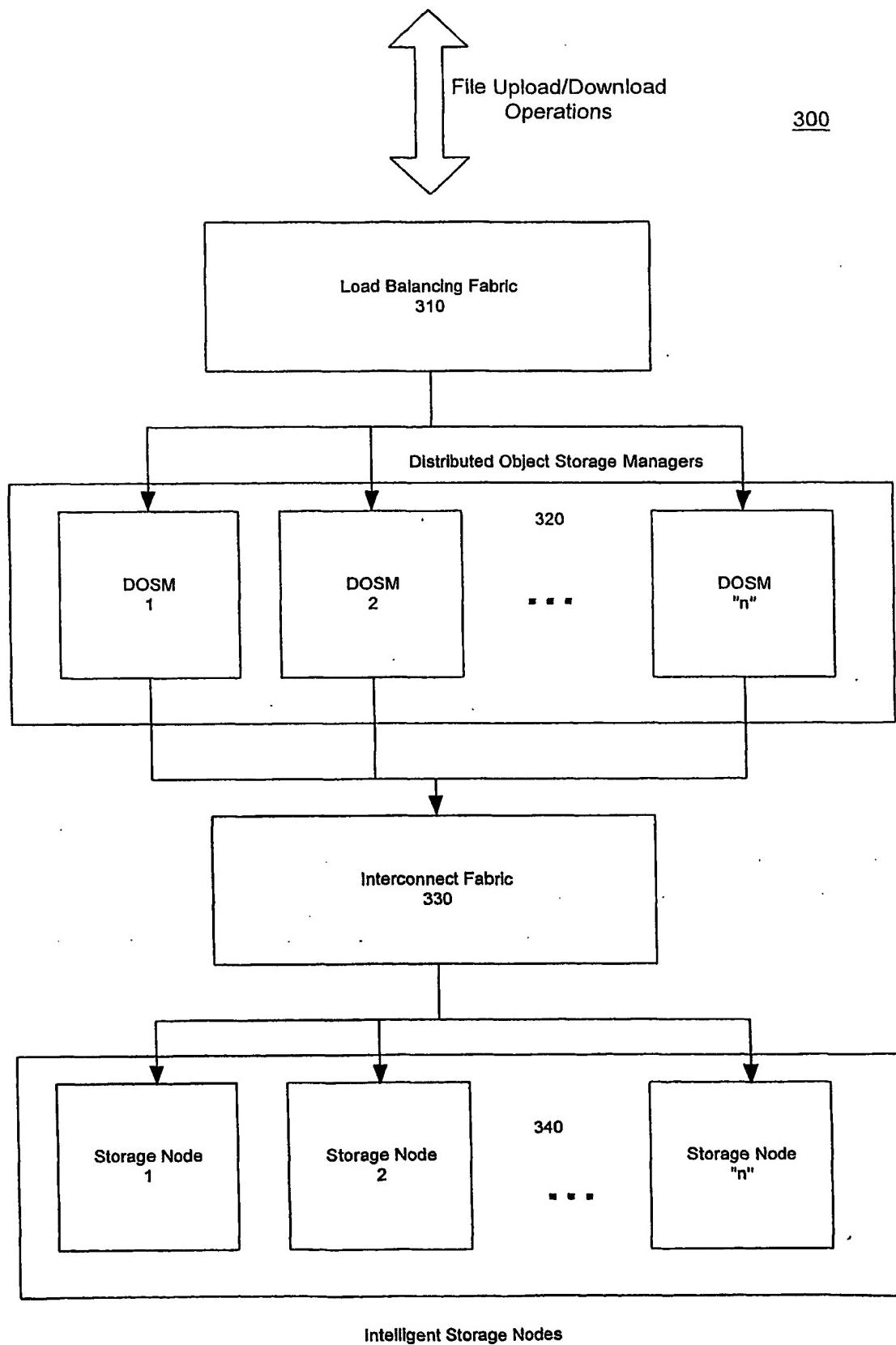
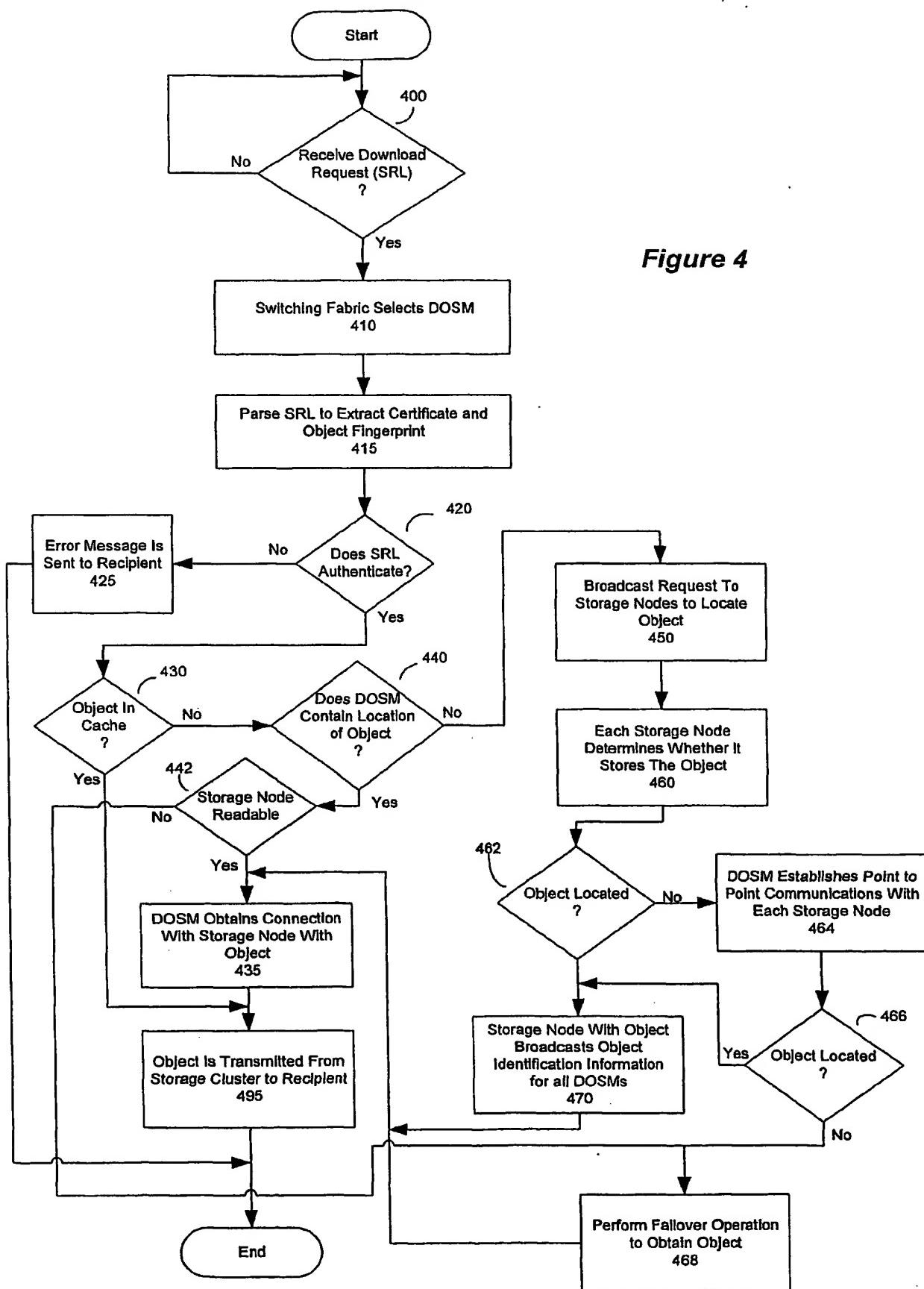


Figure 2

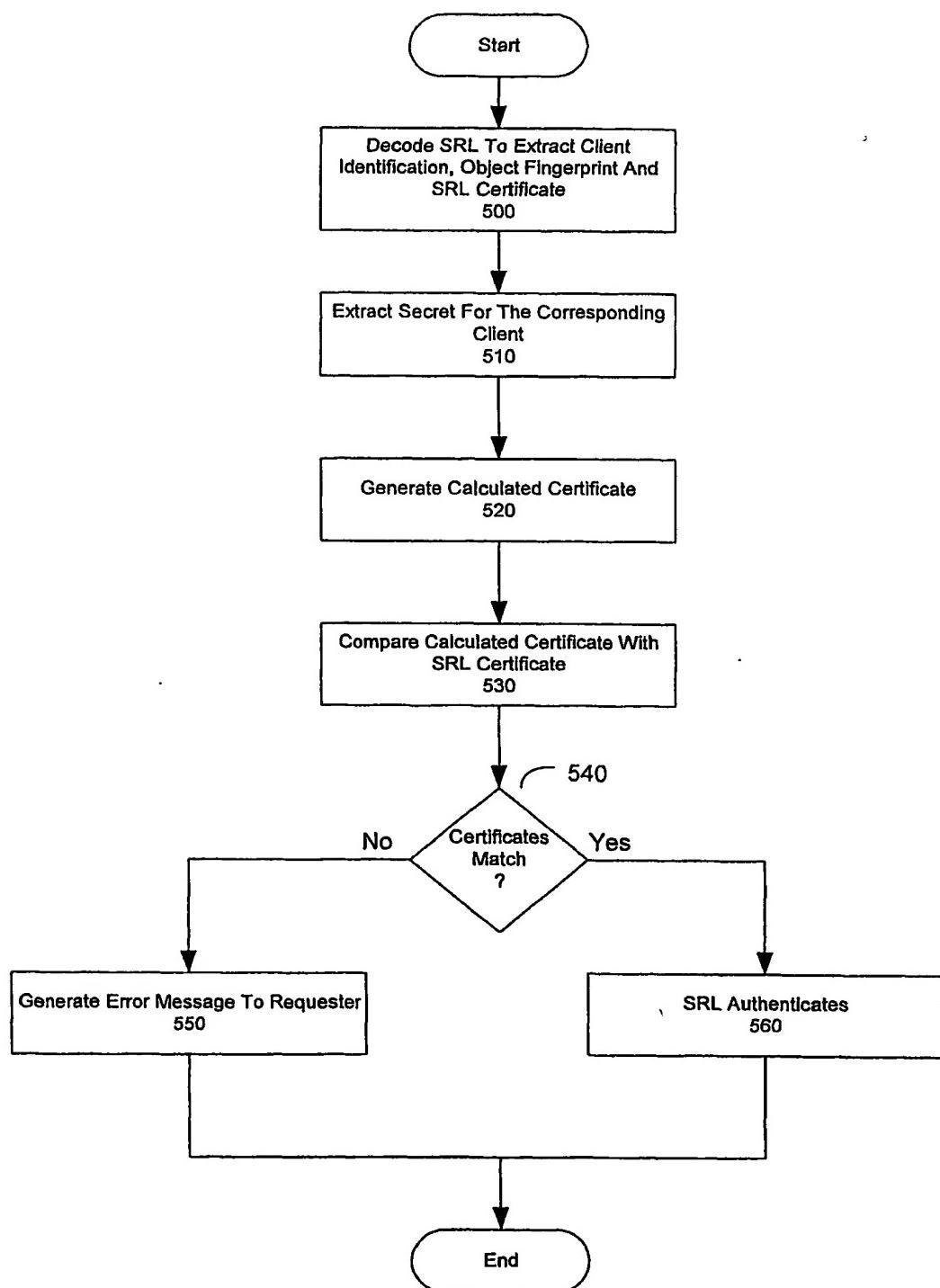
3/32

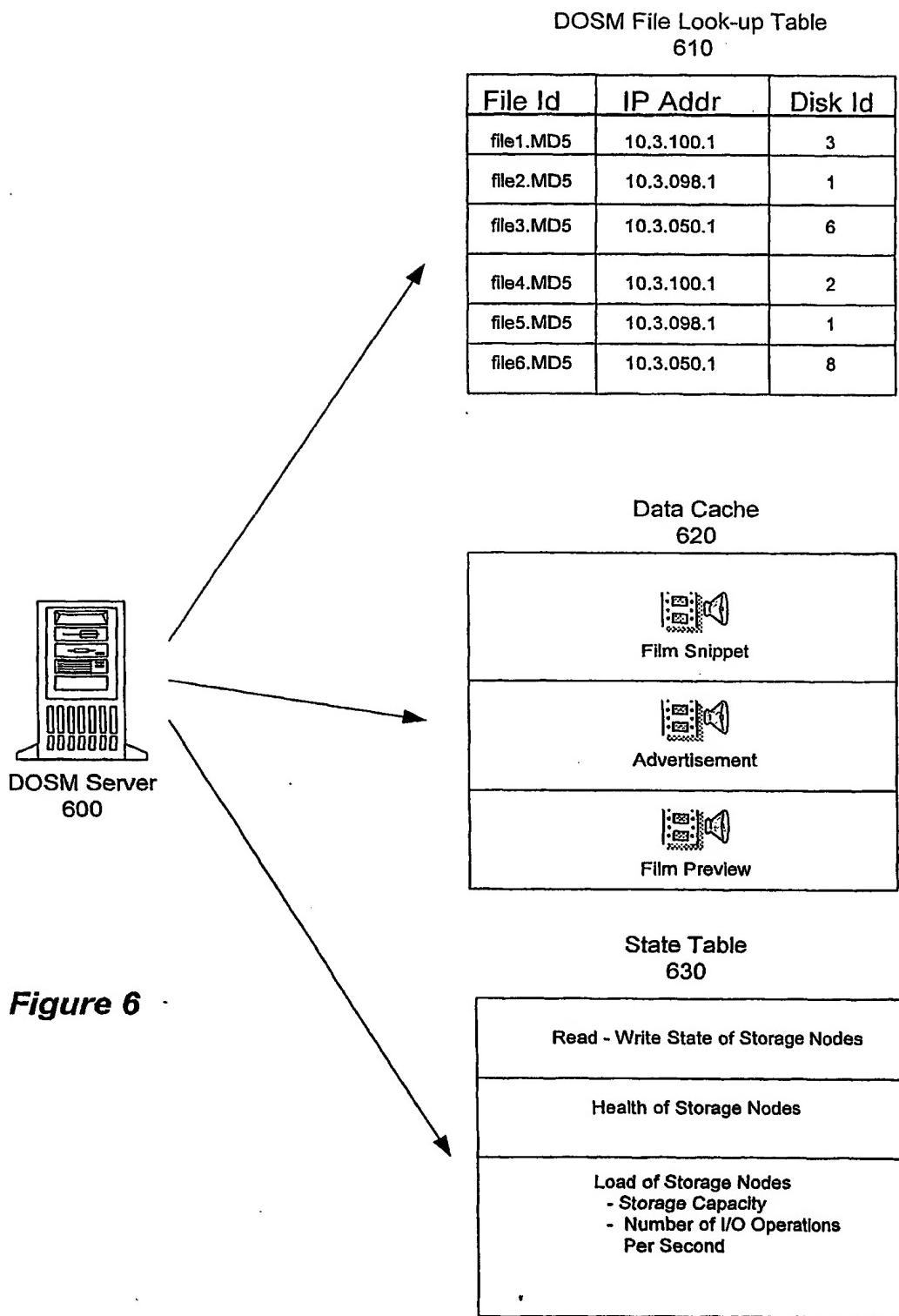
**Figure 3**

4/32



5/32

**Figure 5**

**Figure 6**

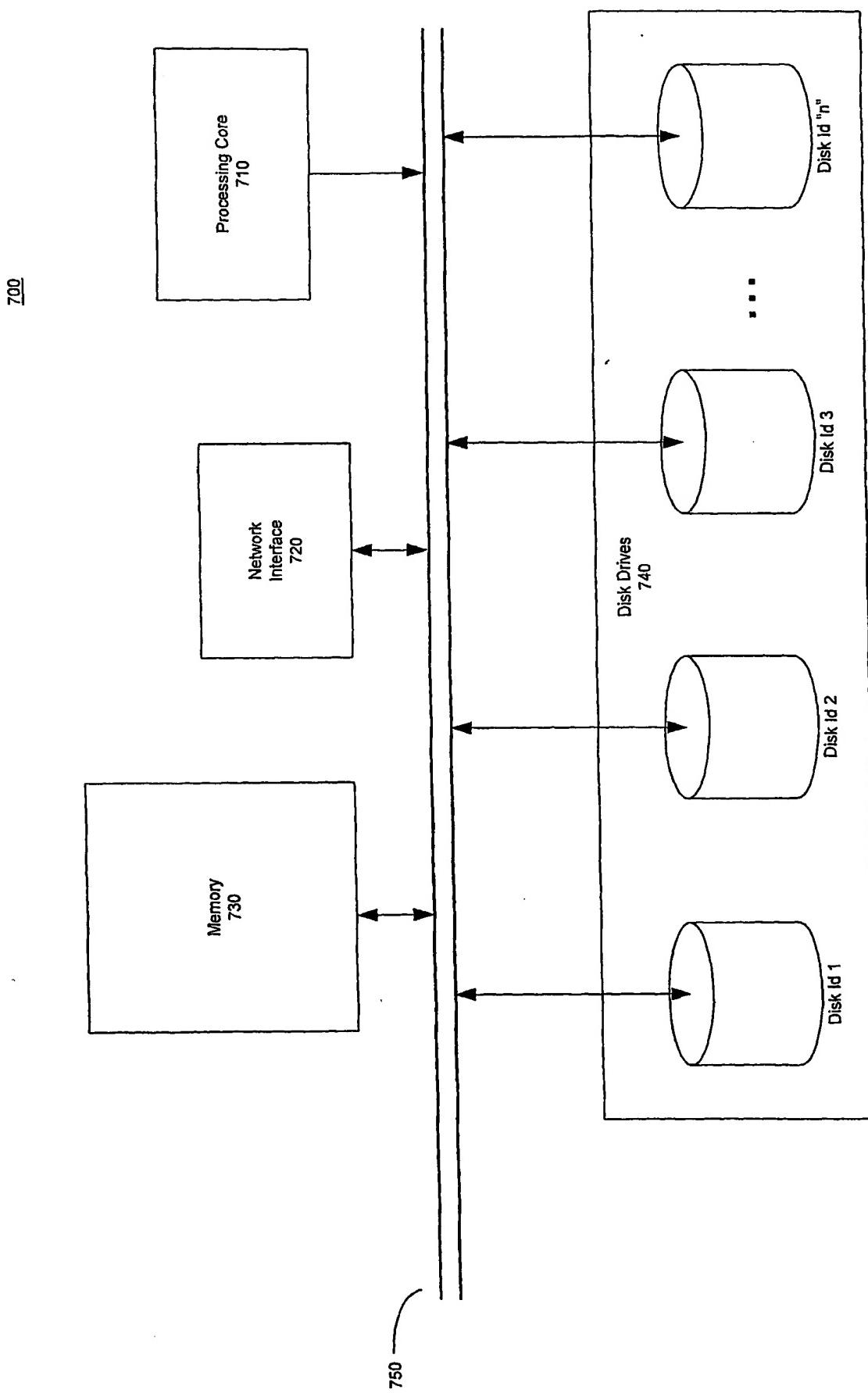
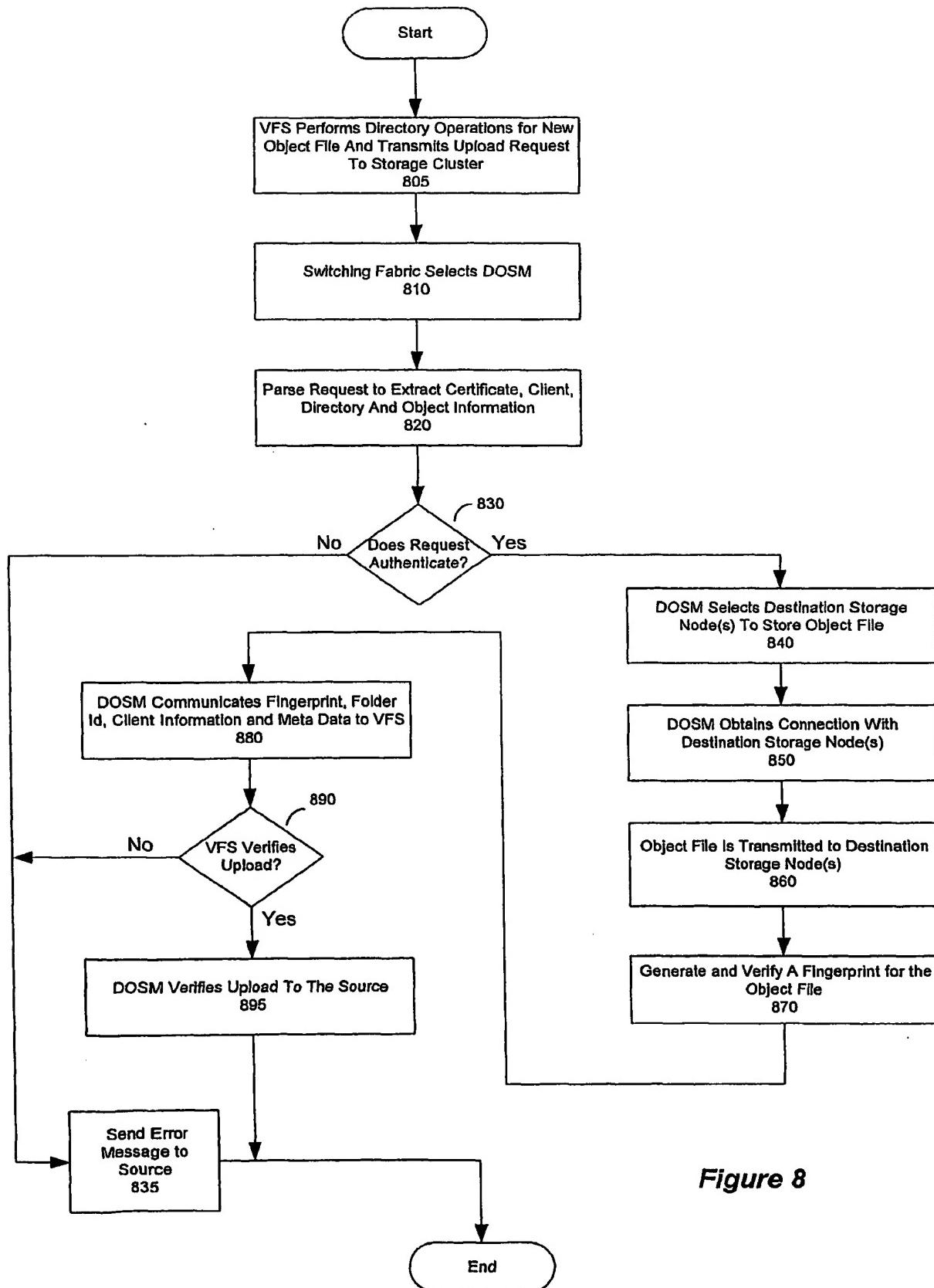


Figure 7

8/32

**Figure 8**

9/32

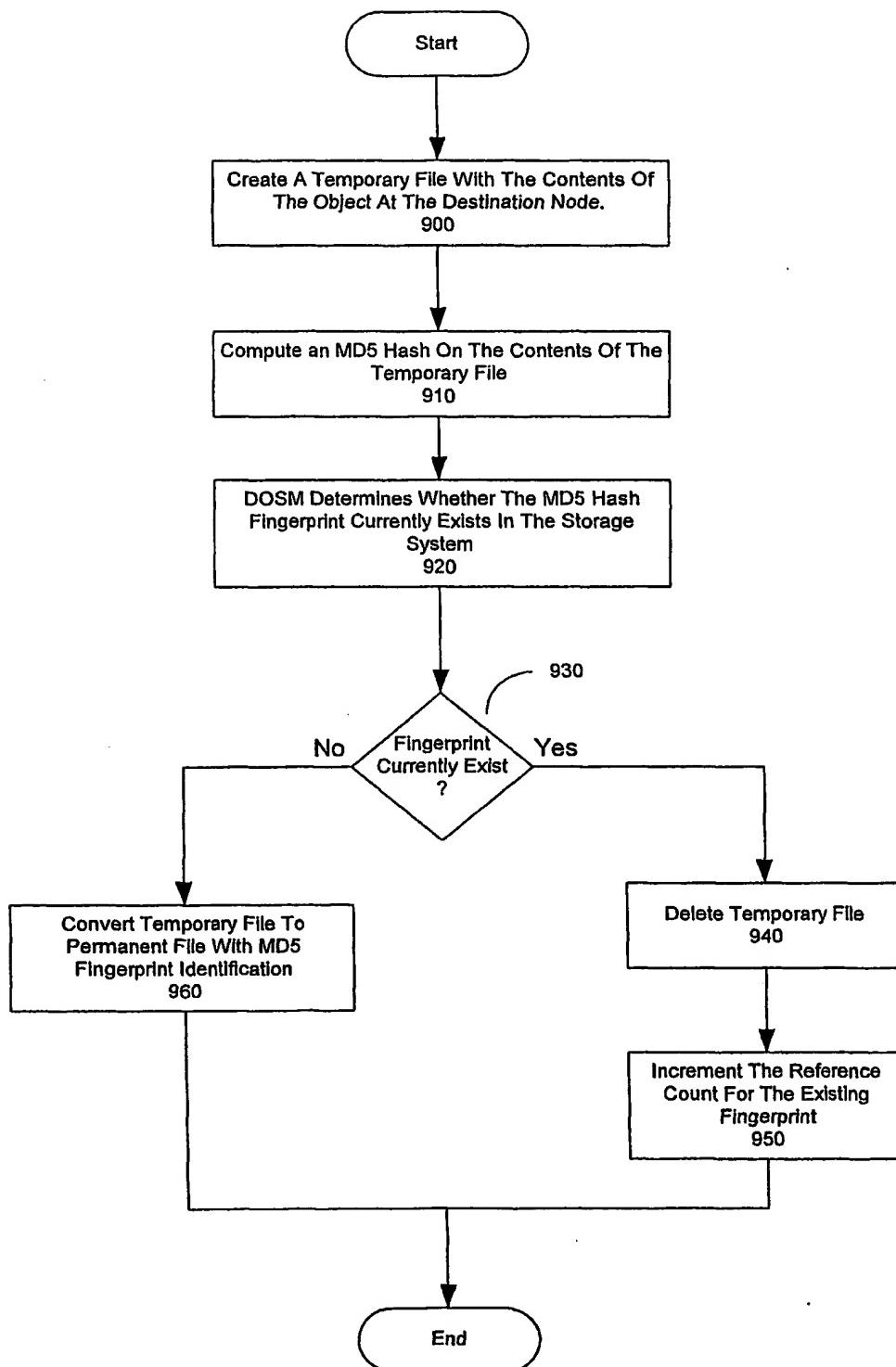
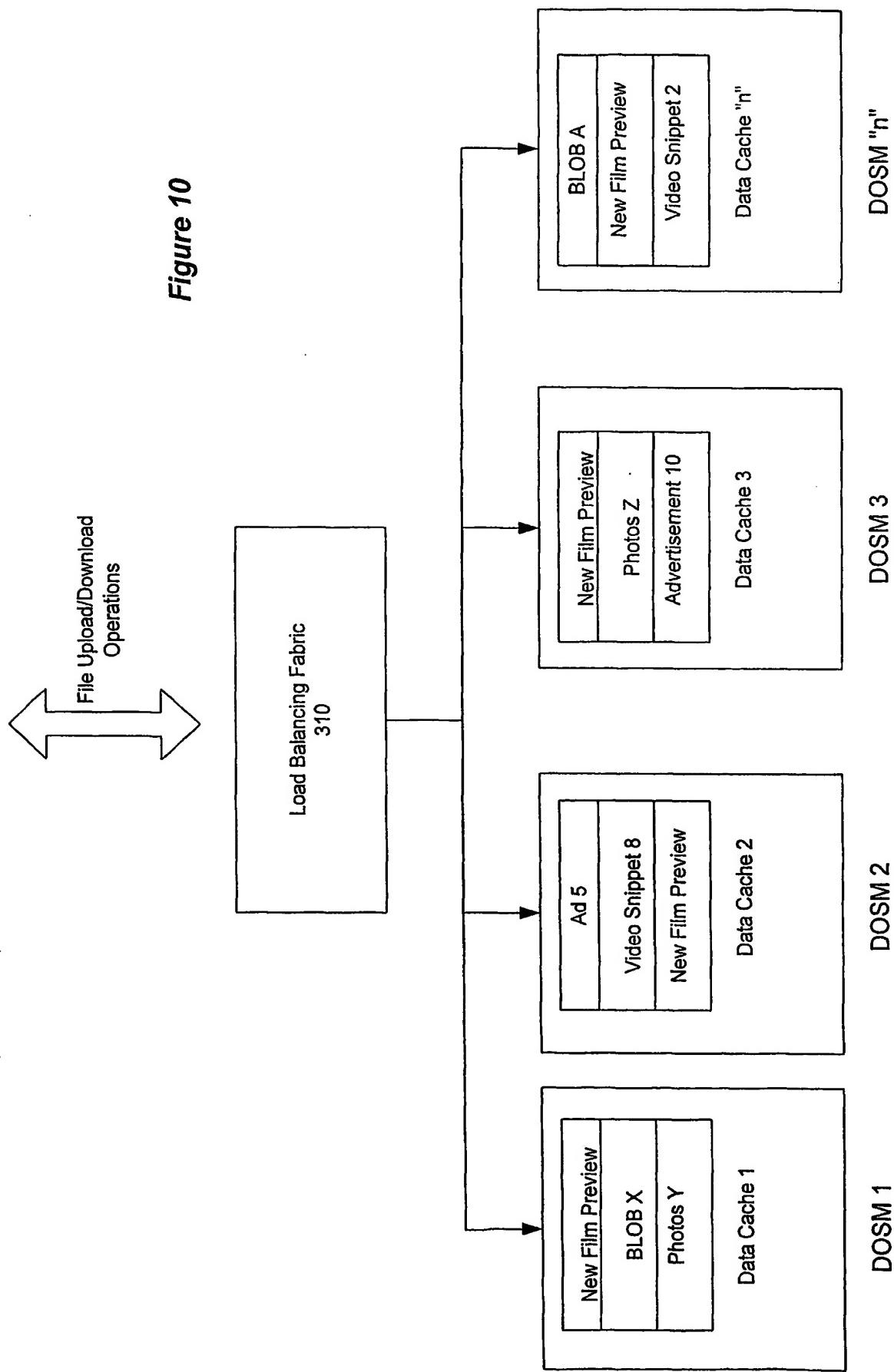
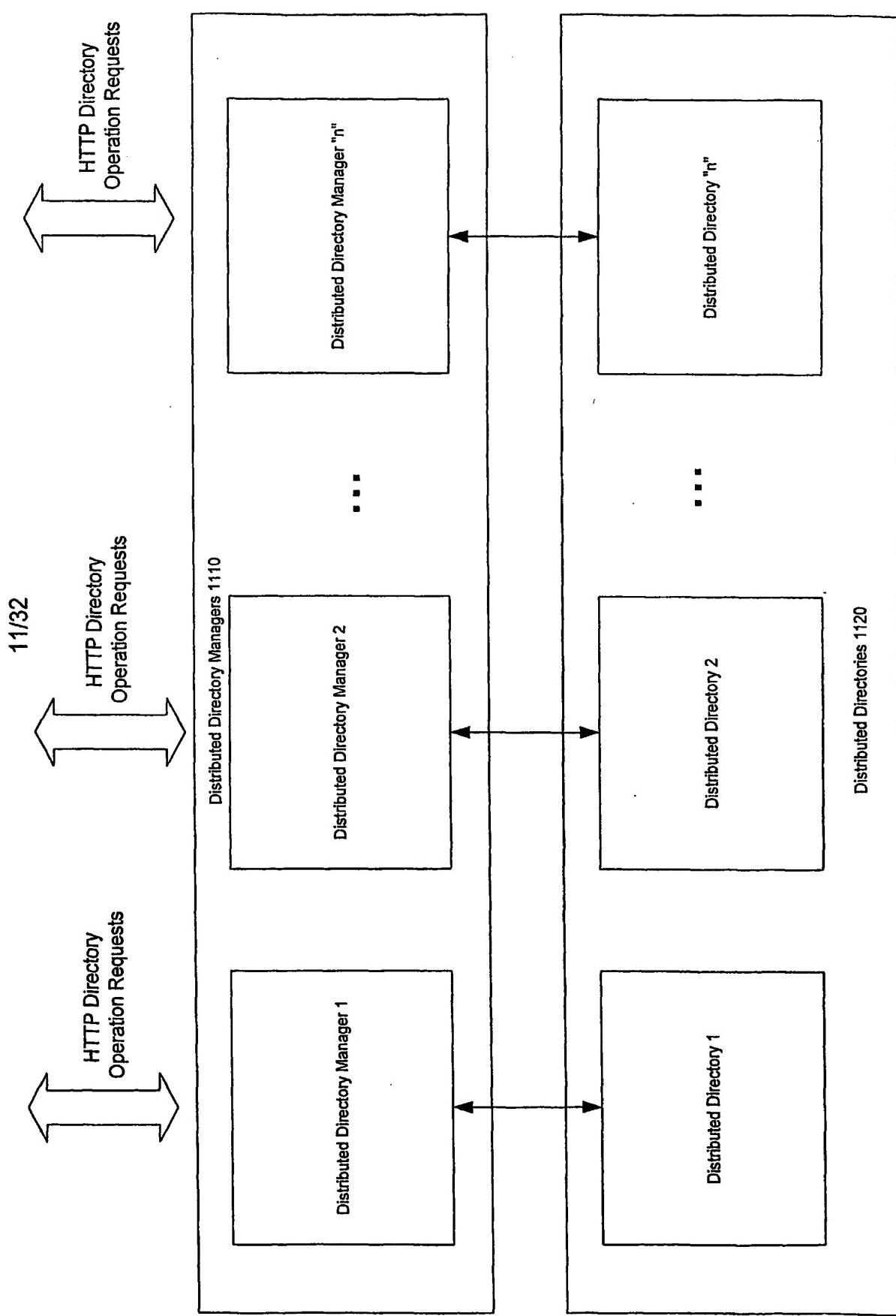
**Figure 9**

Figure 10

11/32



1100

Figure 11

12/32

Customer Table

	Customer Name	Customer Reserved Fields
	Customer A	[Customer stores data ...]
	Customer B	[Customer stores data ...]
	Customer C	[Customer stores data ...]
	Customer D	[Customer stores data ...]

1200

Folder Table

Customer Id	Folder Id	Folder Parent Id	Metadata
3	2	-	[Reserved]
3	100	2	[Reserved]
3	251	2	[Reserved]
3	166	251	[Reserved]

1210

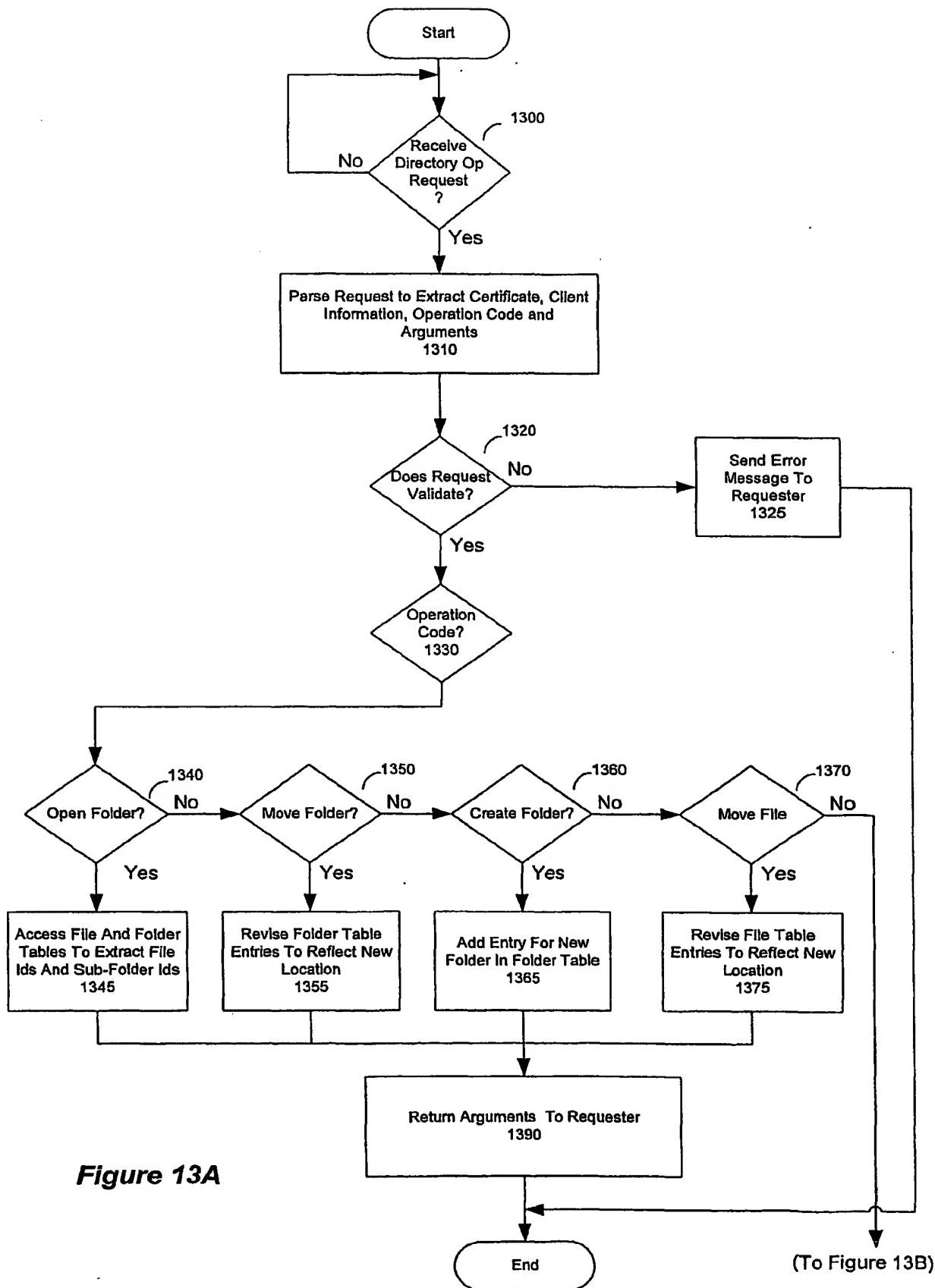
File Table

Customer Id	File Handle	Folder Id	Folder Parent Id	Metadata
3	52.MD5	100	2	[Reserved]
3	55.MD5	100	2	[Reserved]
3	99.MD5	166	251	[Reserved]
3	67.MD5	166	251	[Reserved]

1220

Figure 12

13/32

**Figure 13A**

(To Figure 13B)

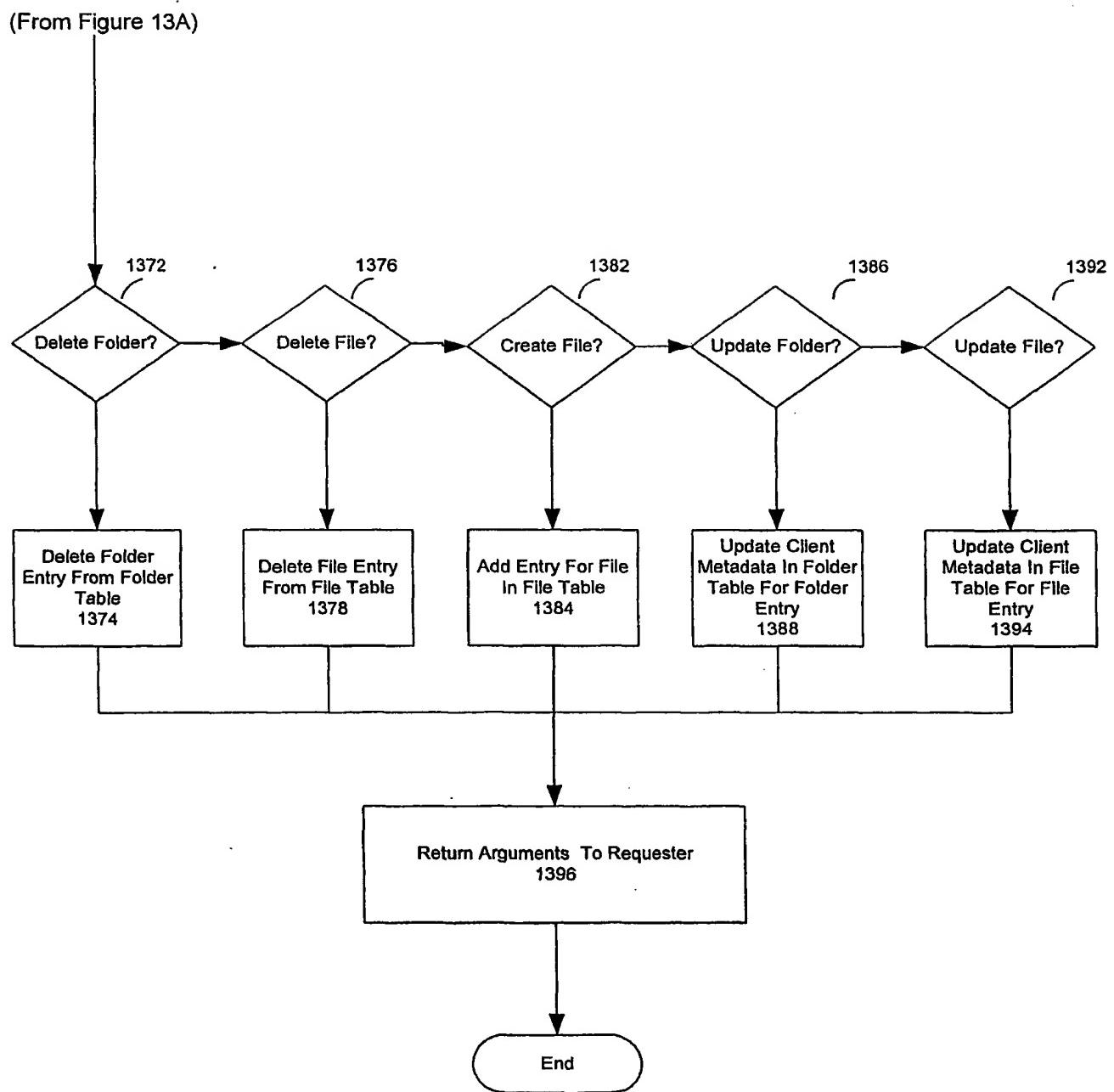


Figure 13B

15/32

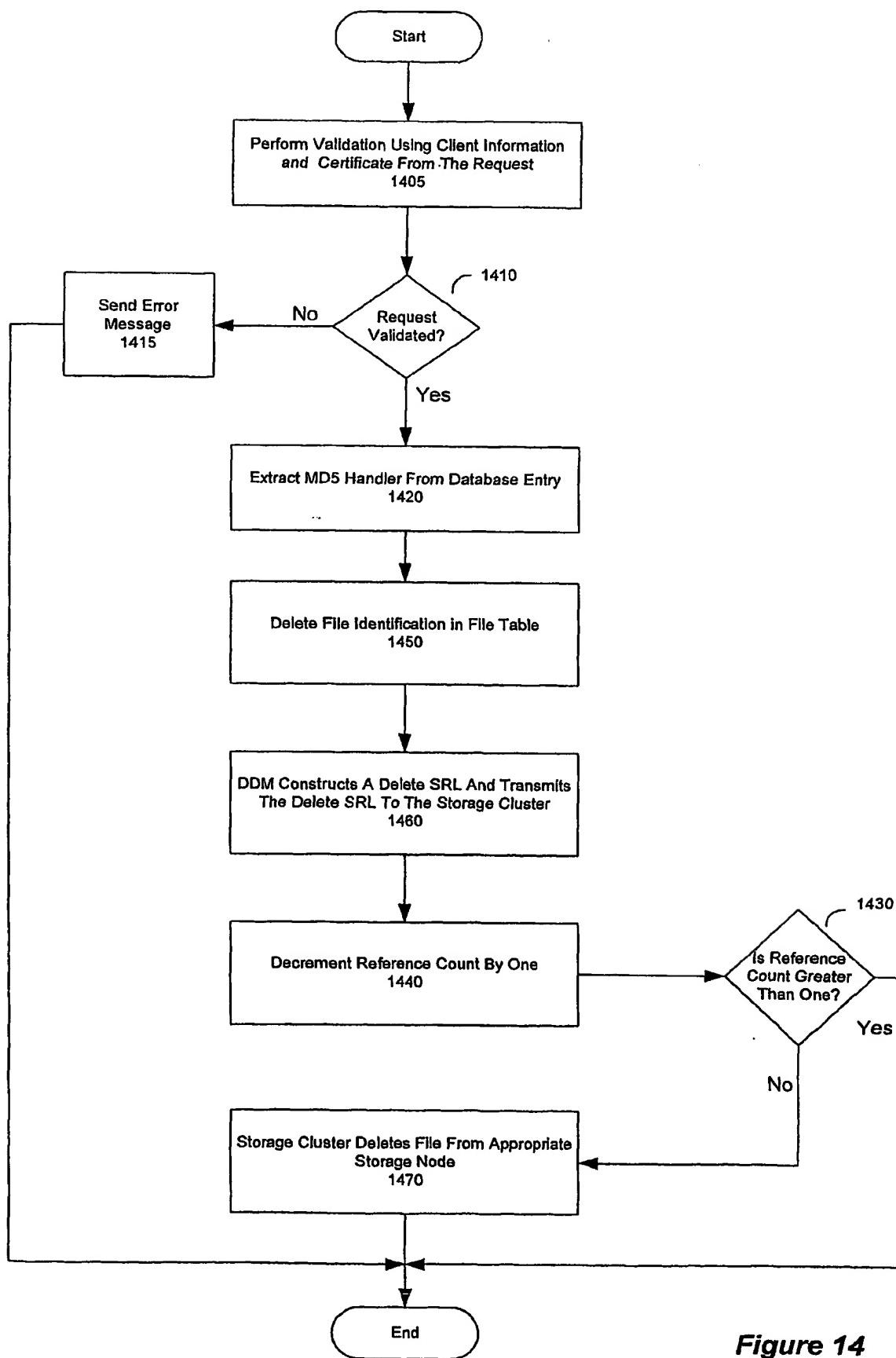


Figure 14

16/32

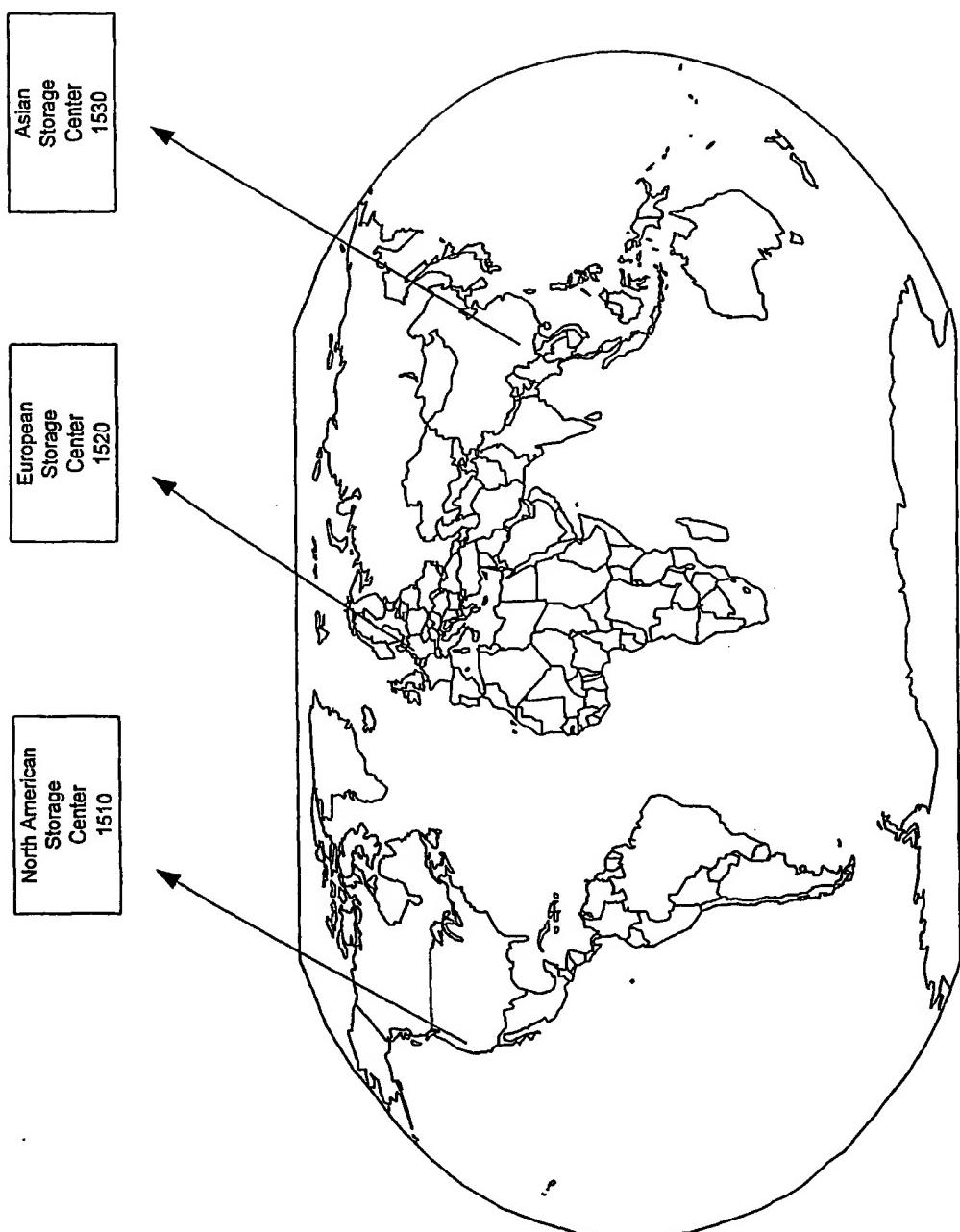
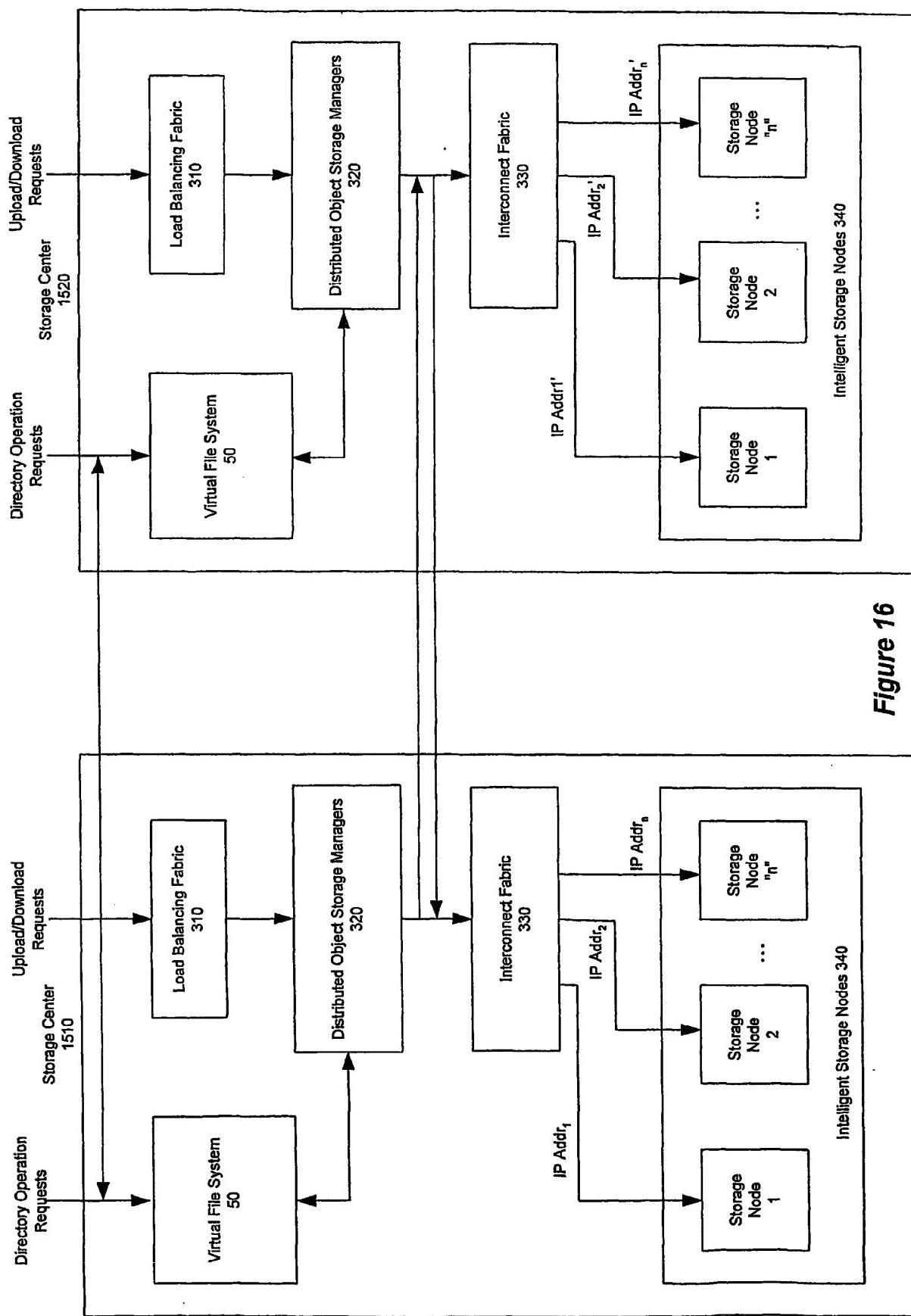


Figure 15

**Figure 16**

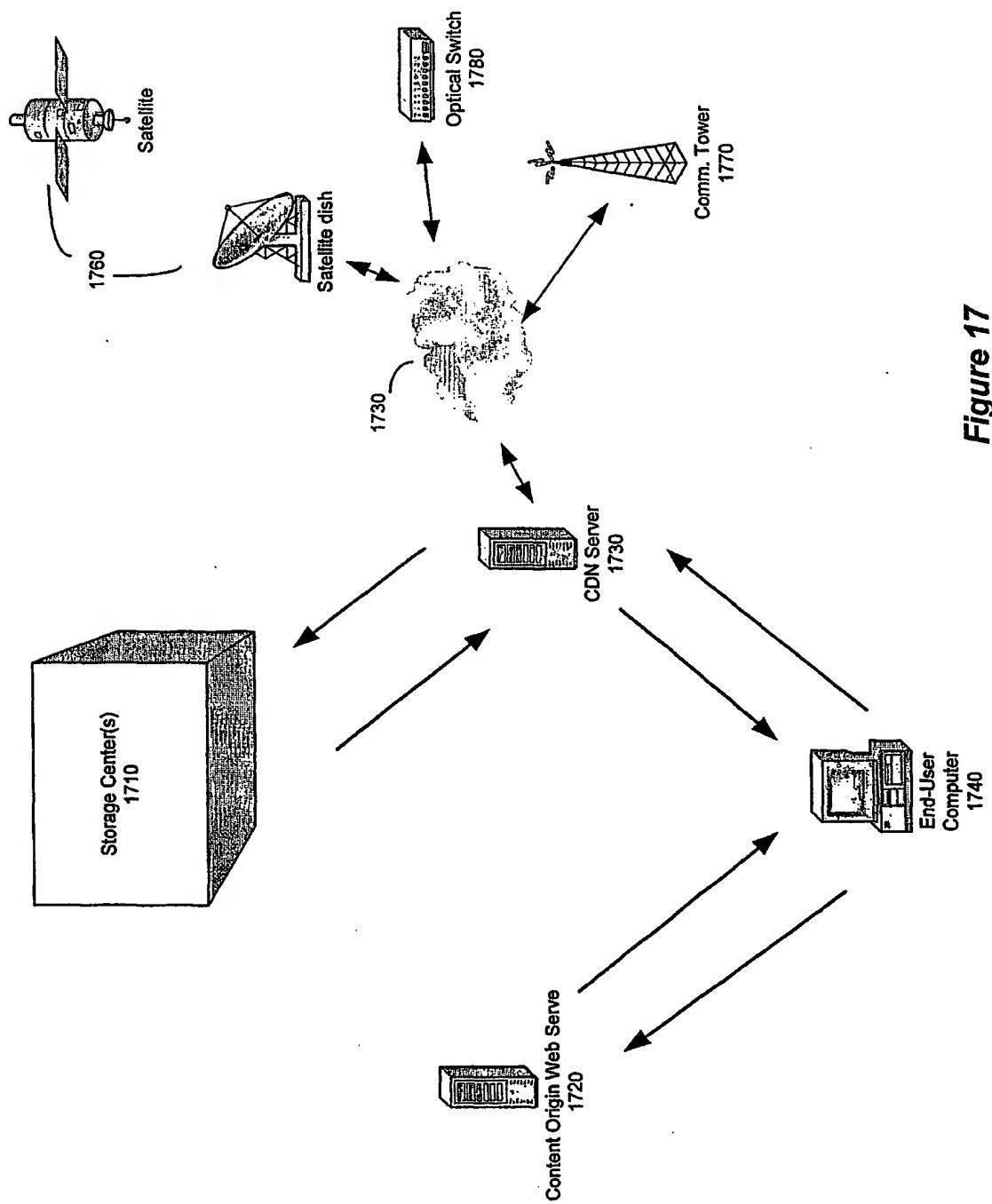
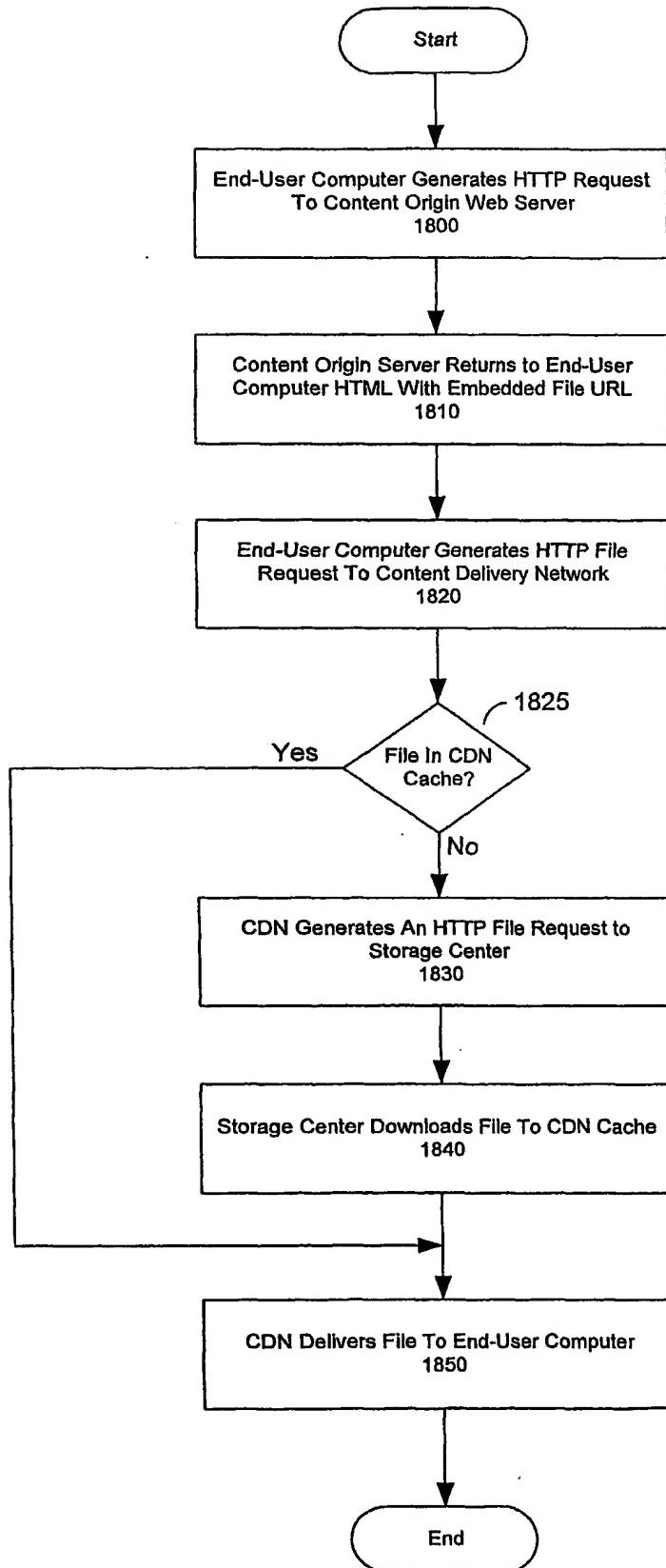


Figure 17

19/32

**Figure 18**

20/32

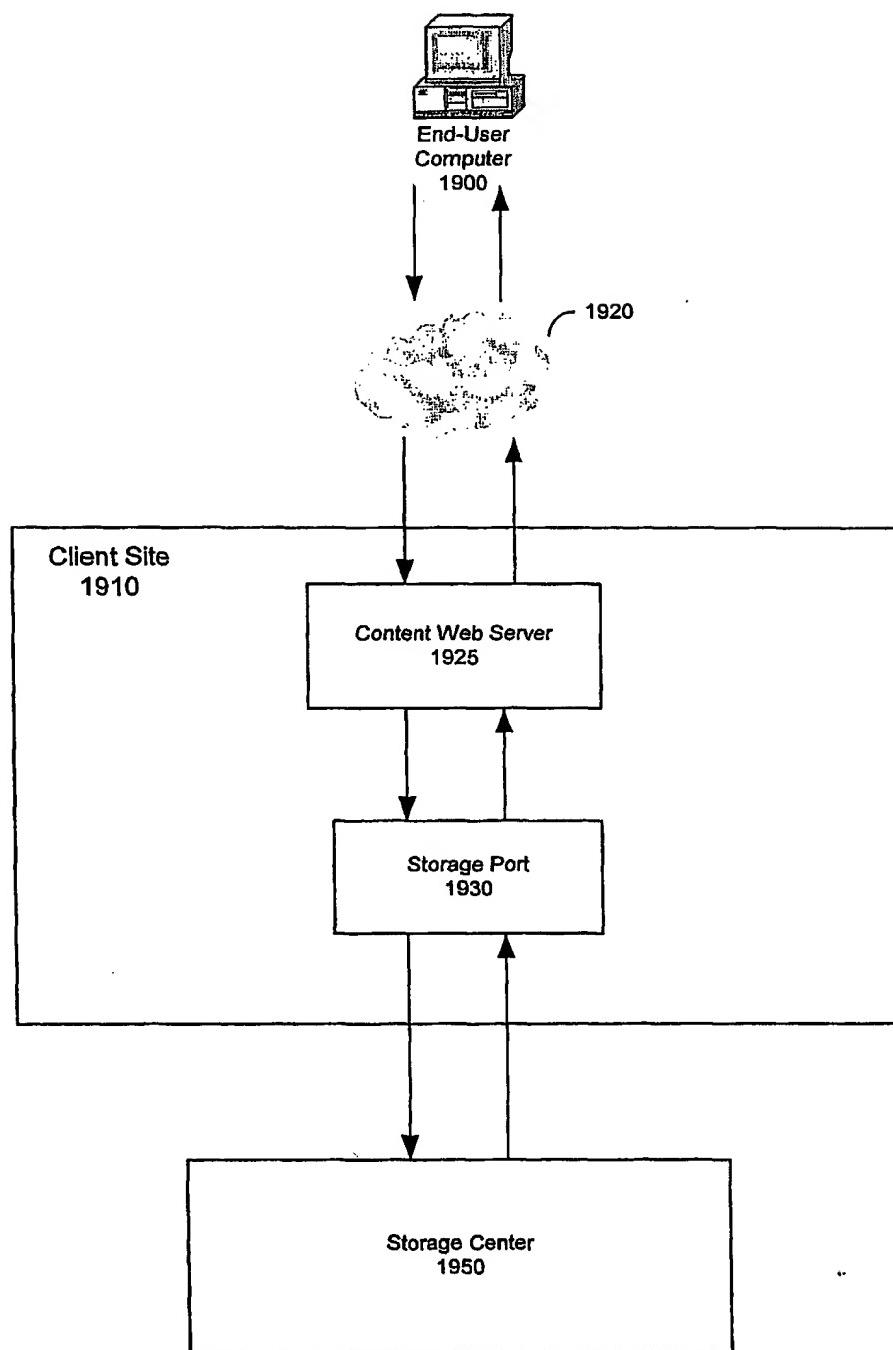
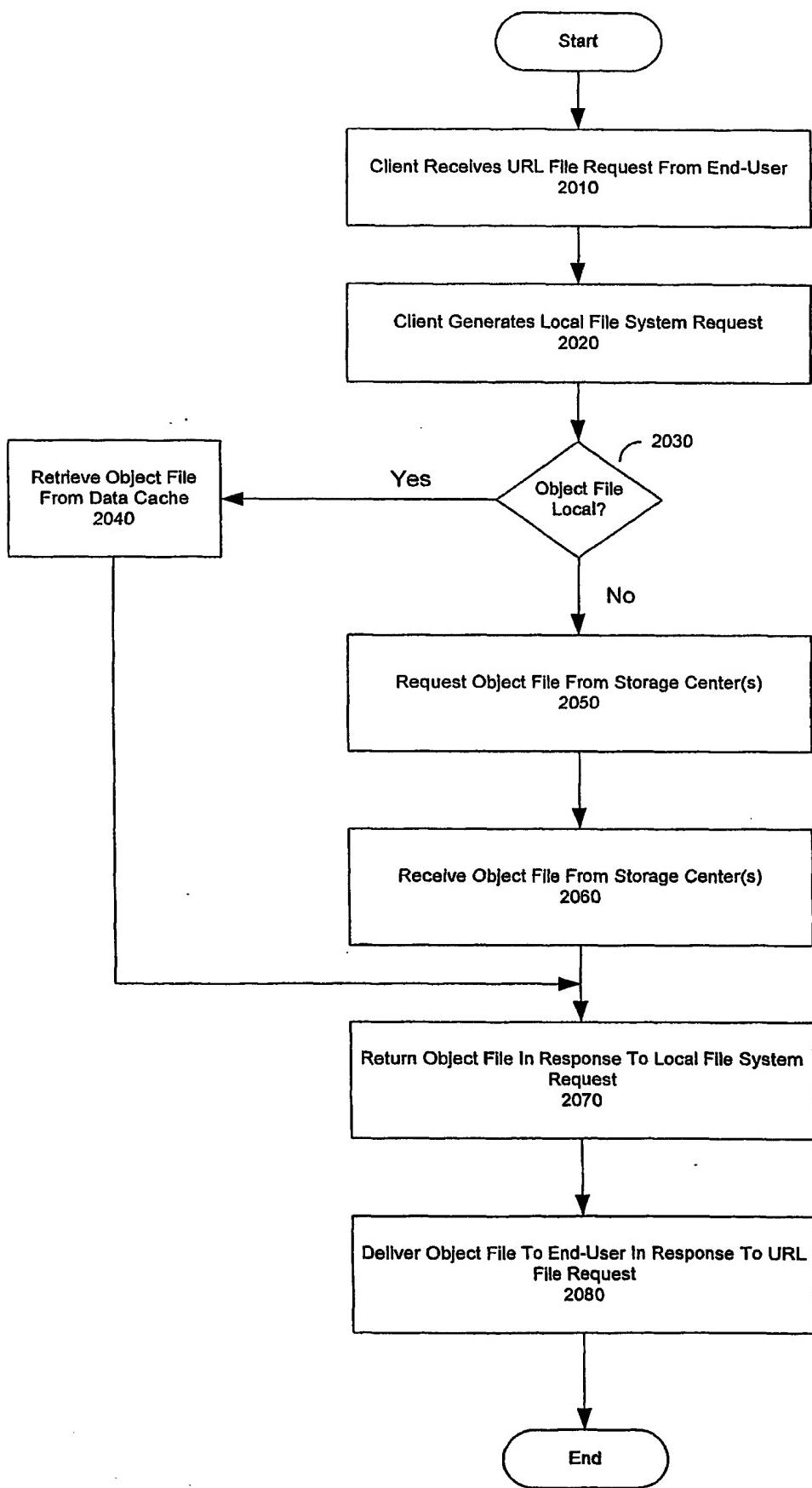


Figure 19

21/32

**Figure 20**

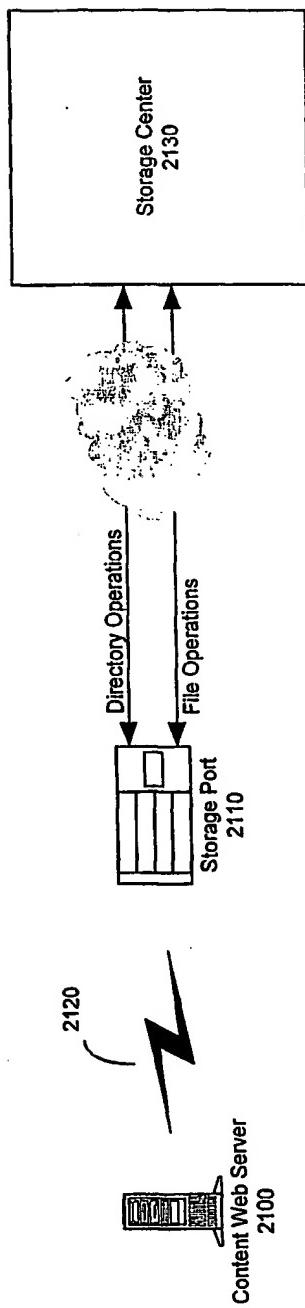


Figure 21a

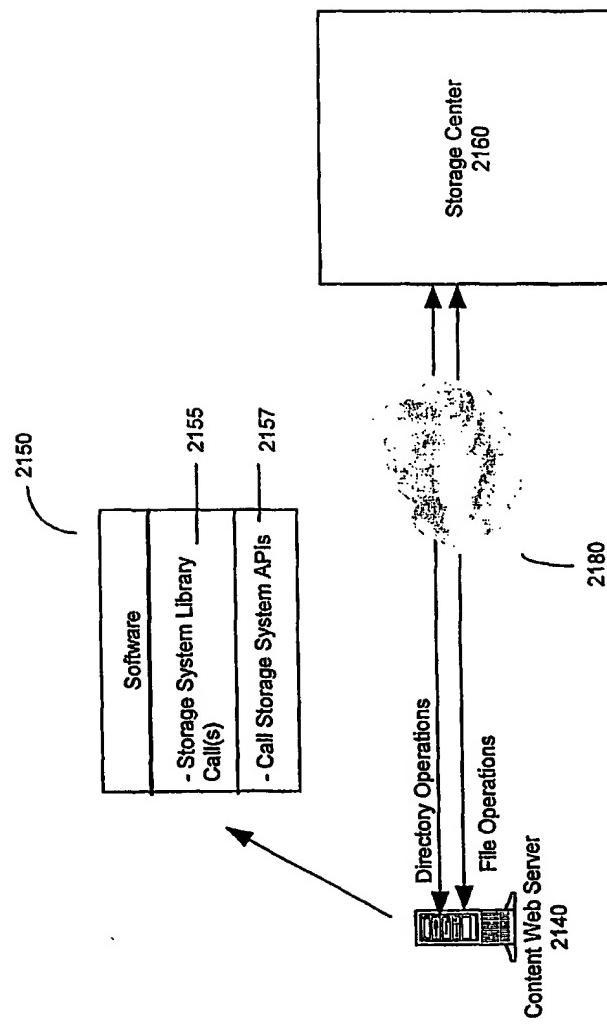


Figure 21b

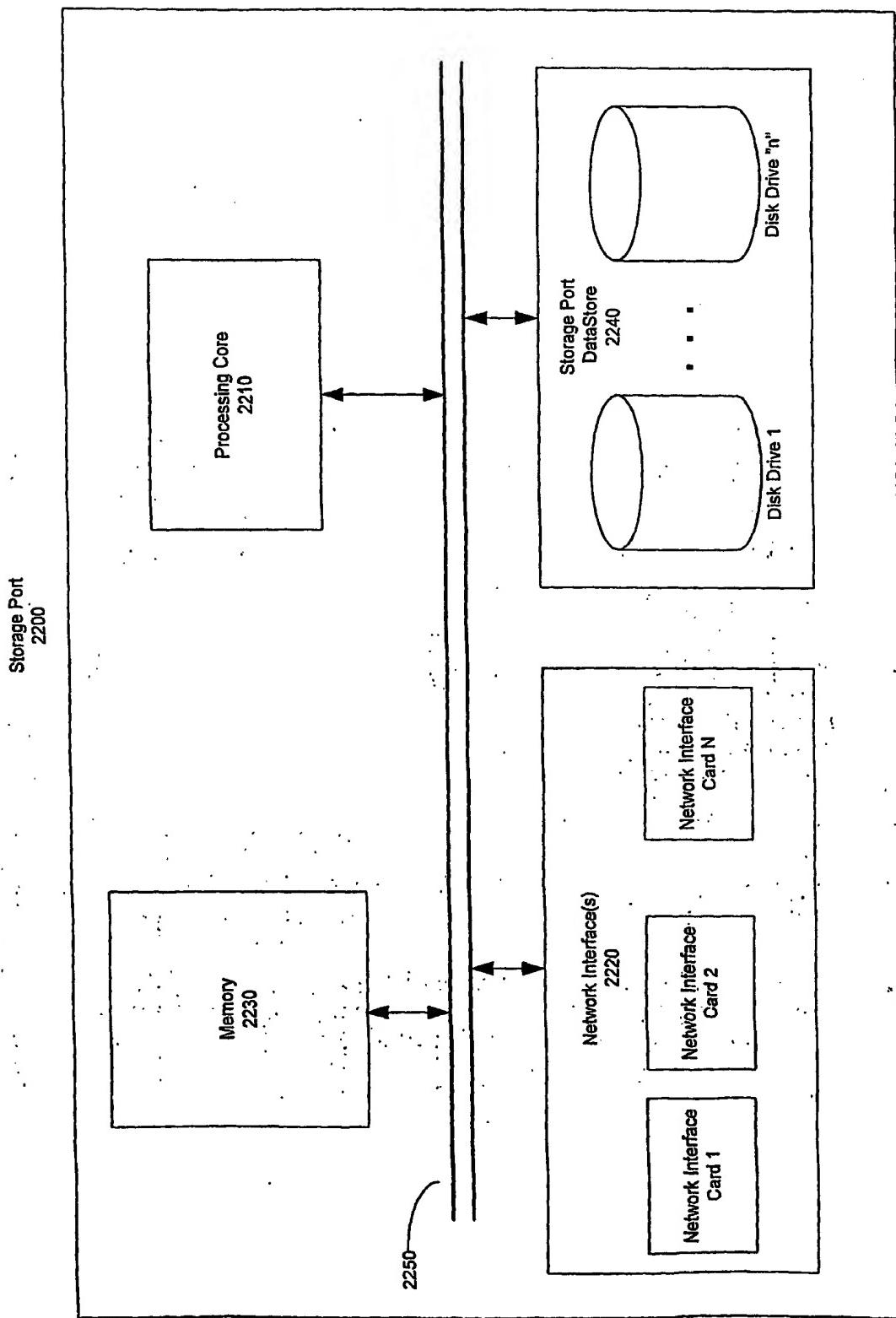
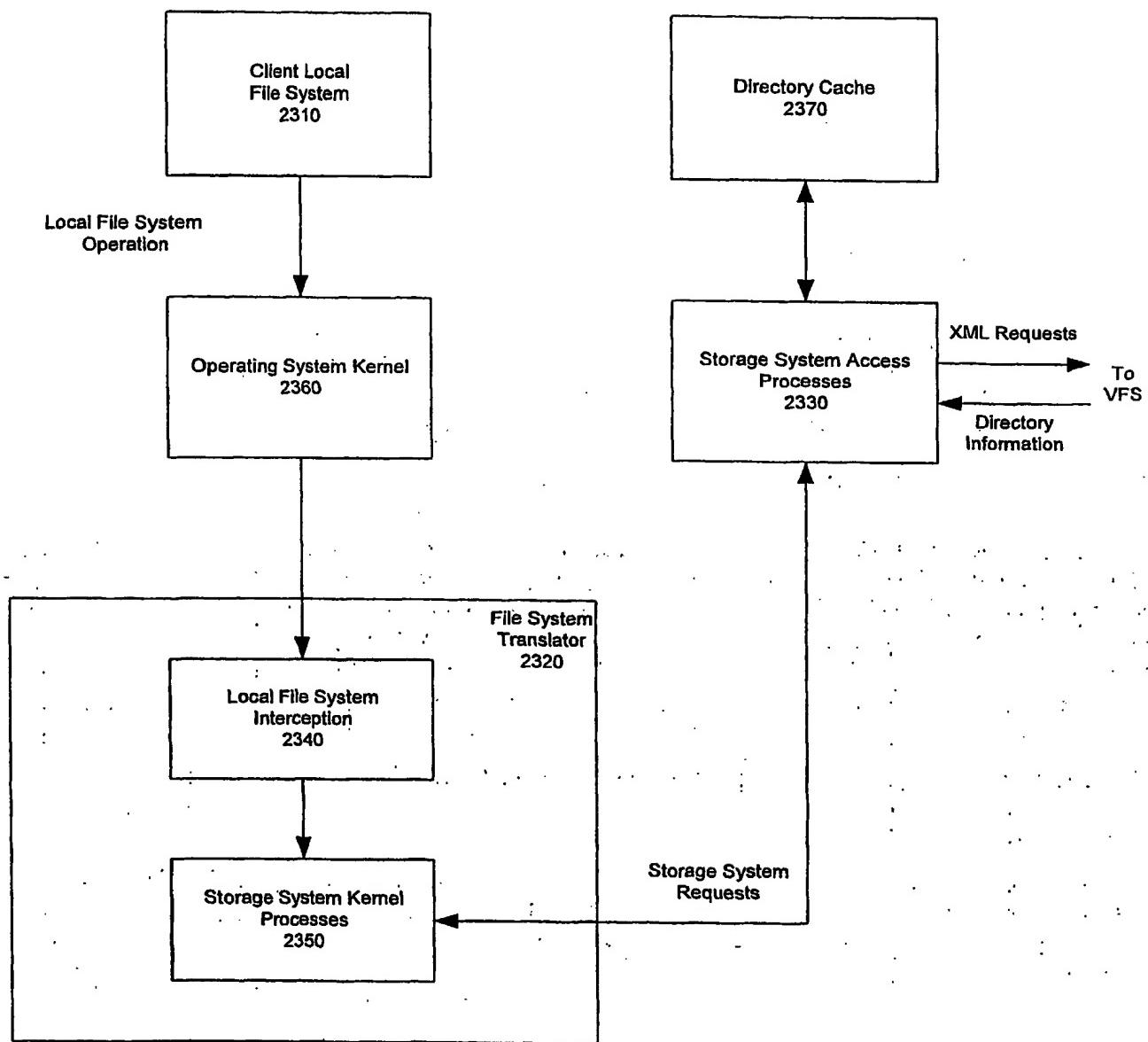
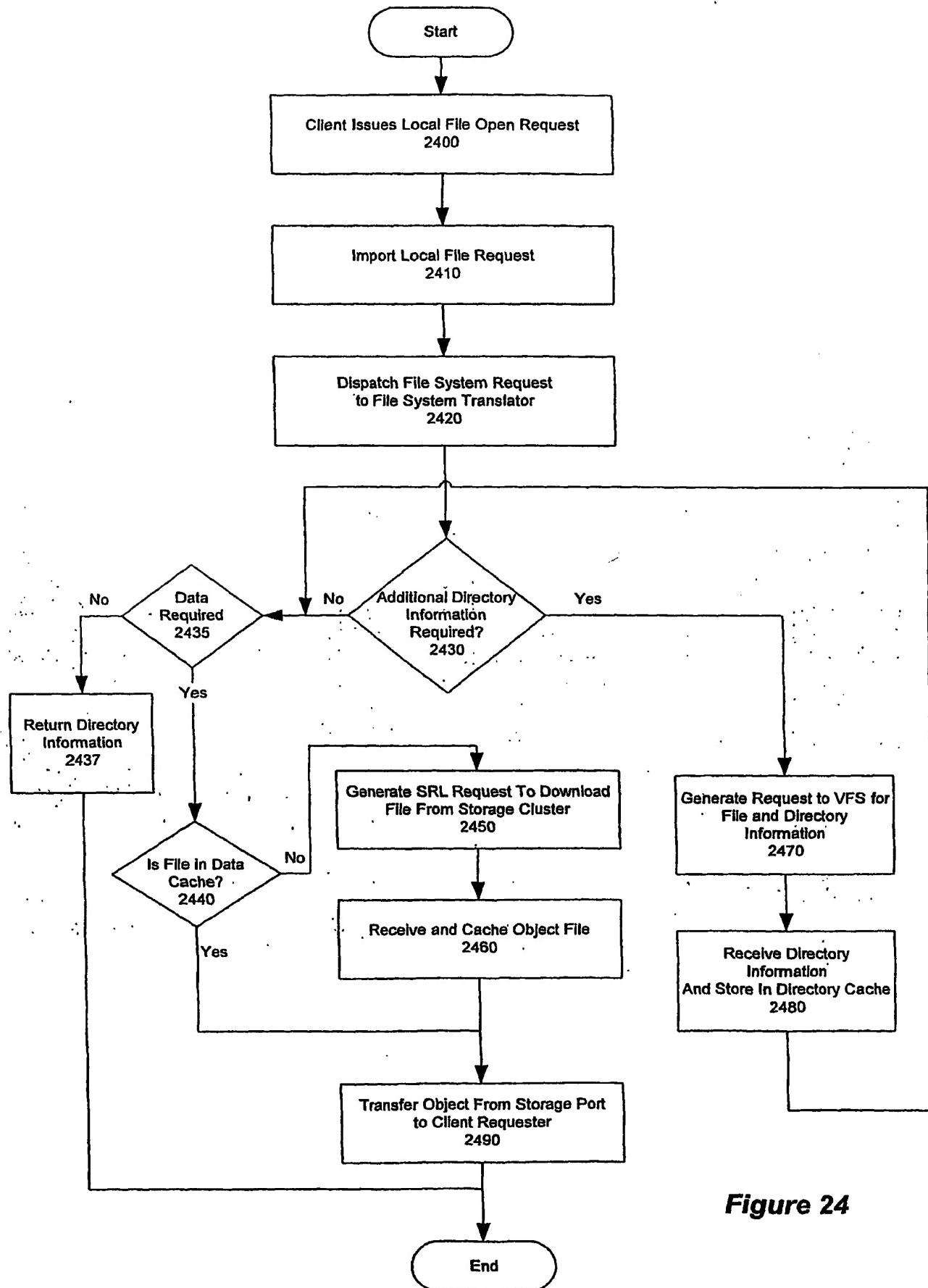


Figure 22

2300**Figure 23**

25/32

**Figure 24**

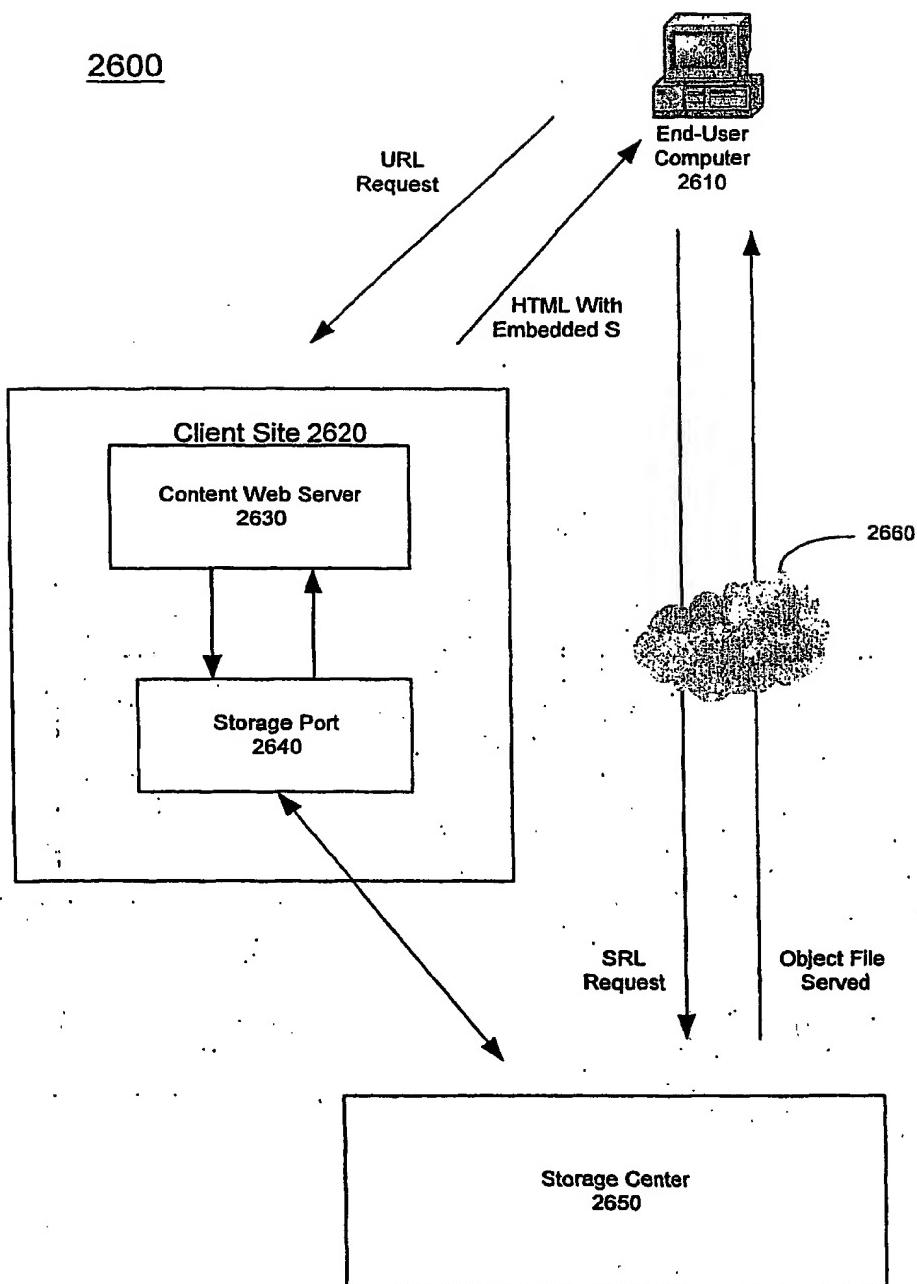


Figure 25

27/32

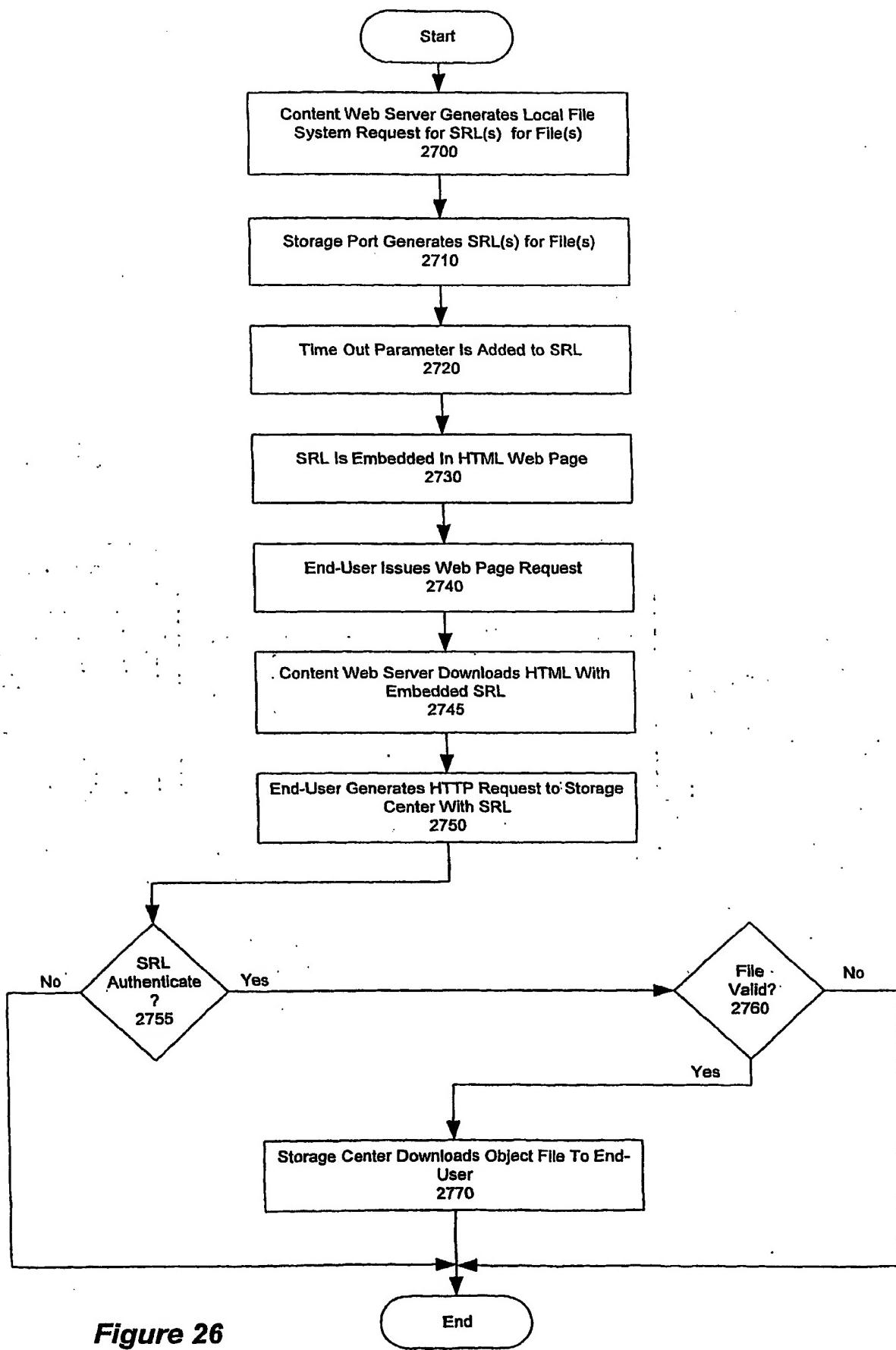


Figure 26

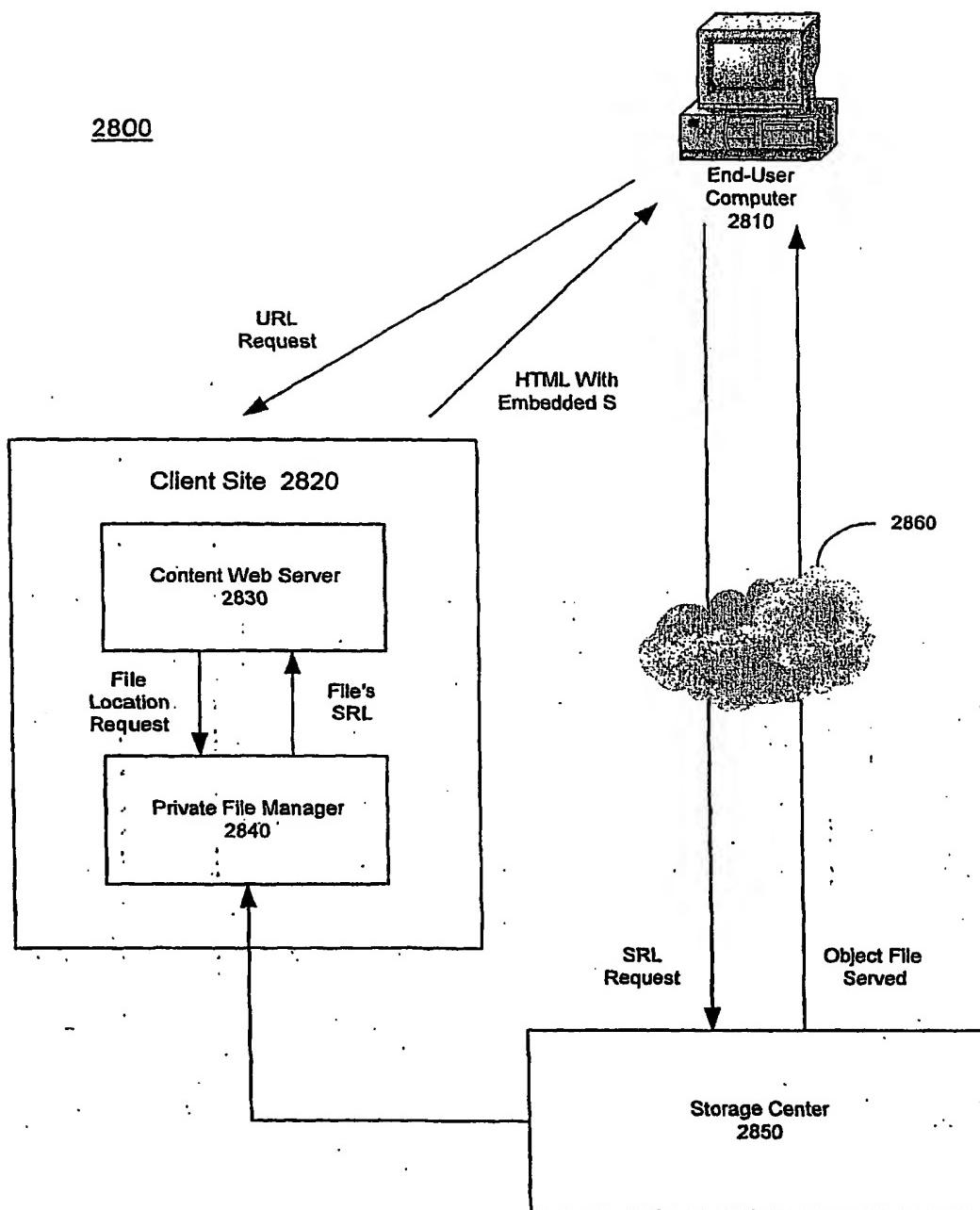


Figure 27

29/32

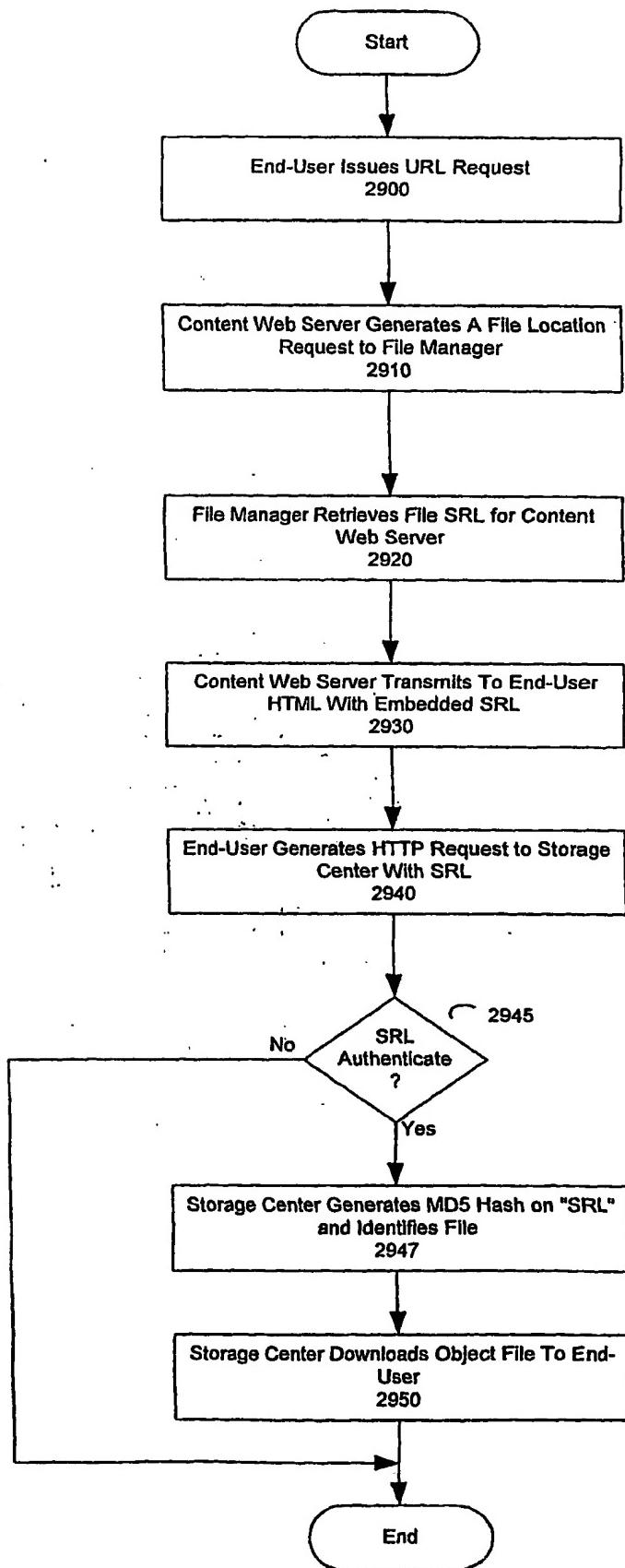


Figure 28

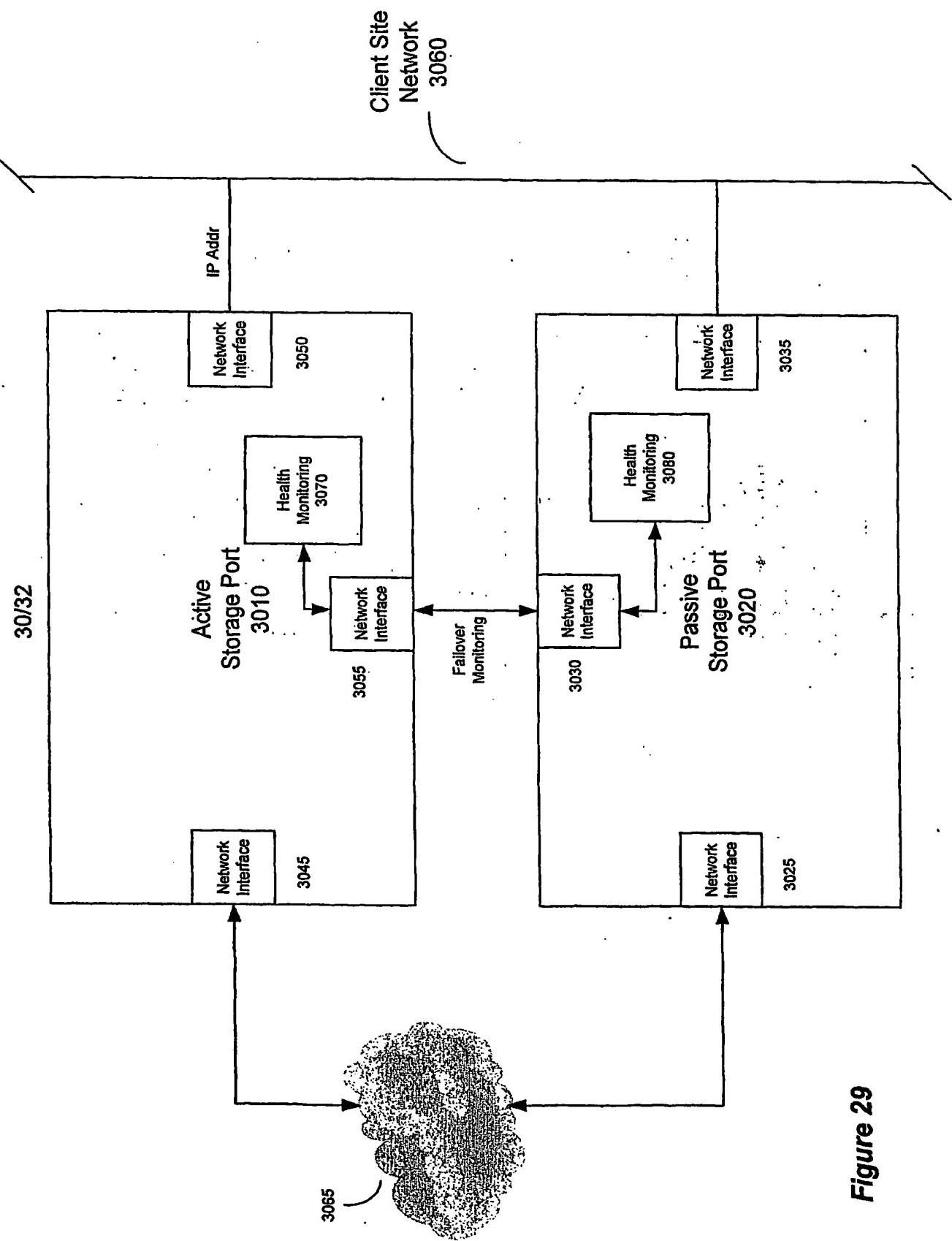


Figure 29

31/32

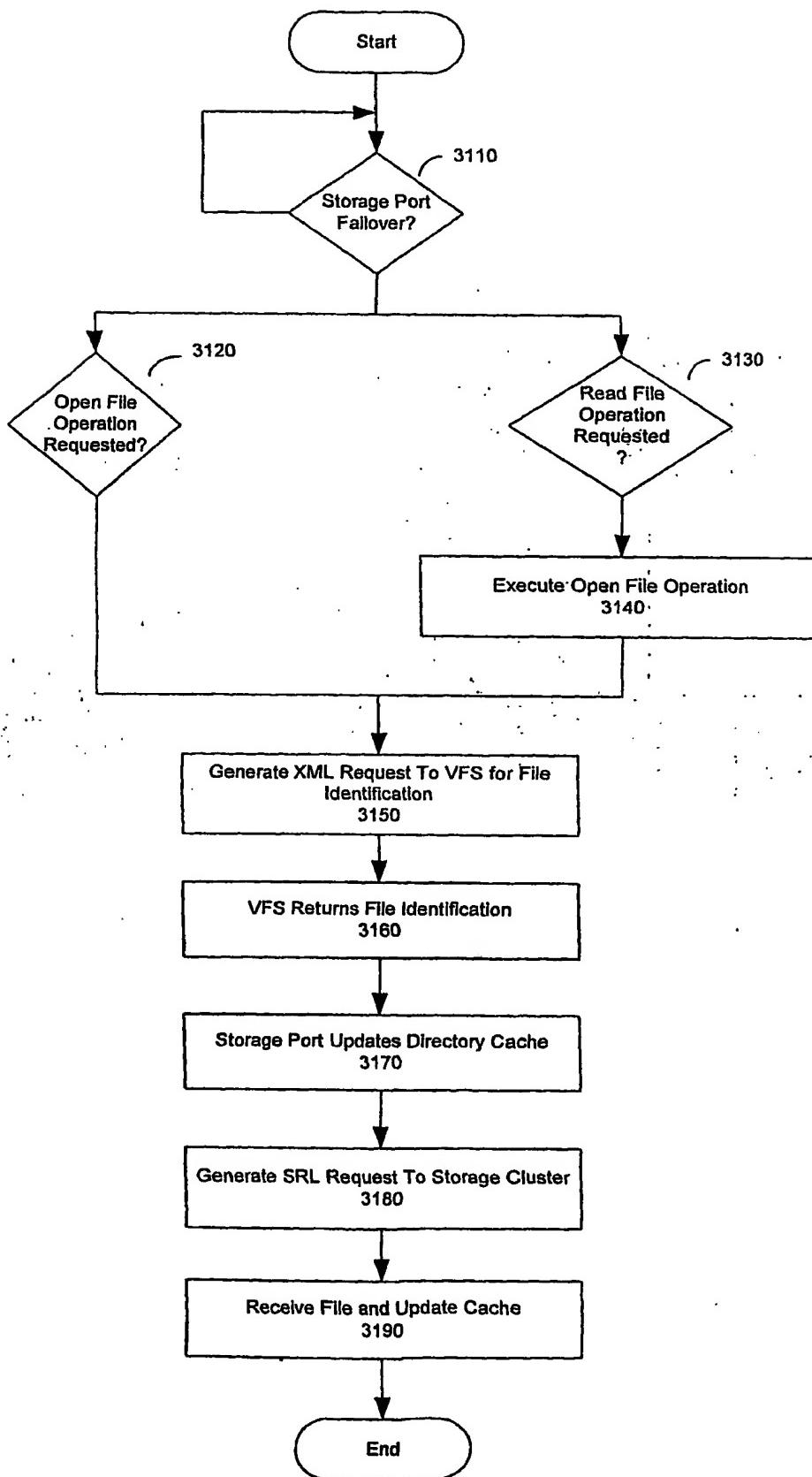
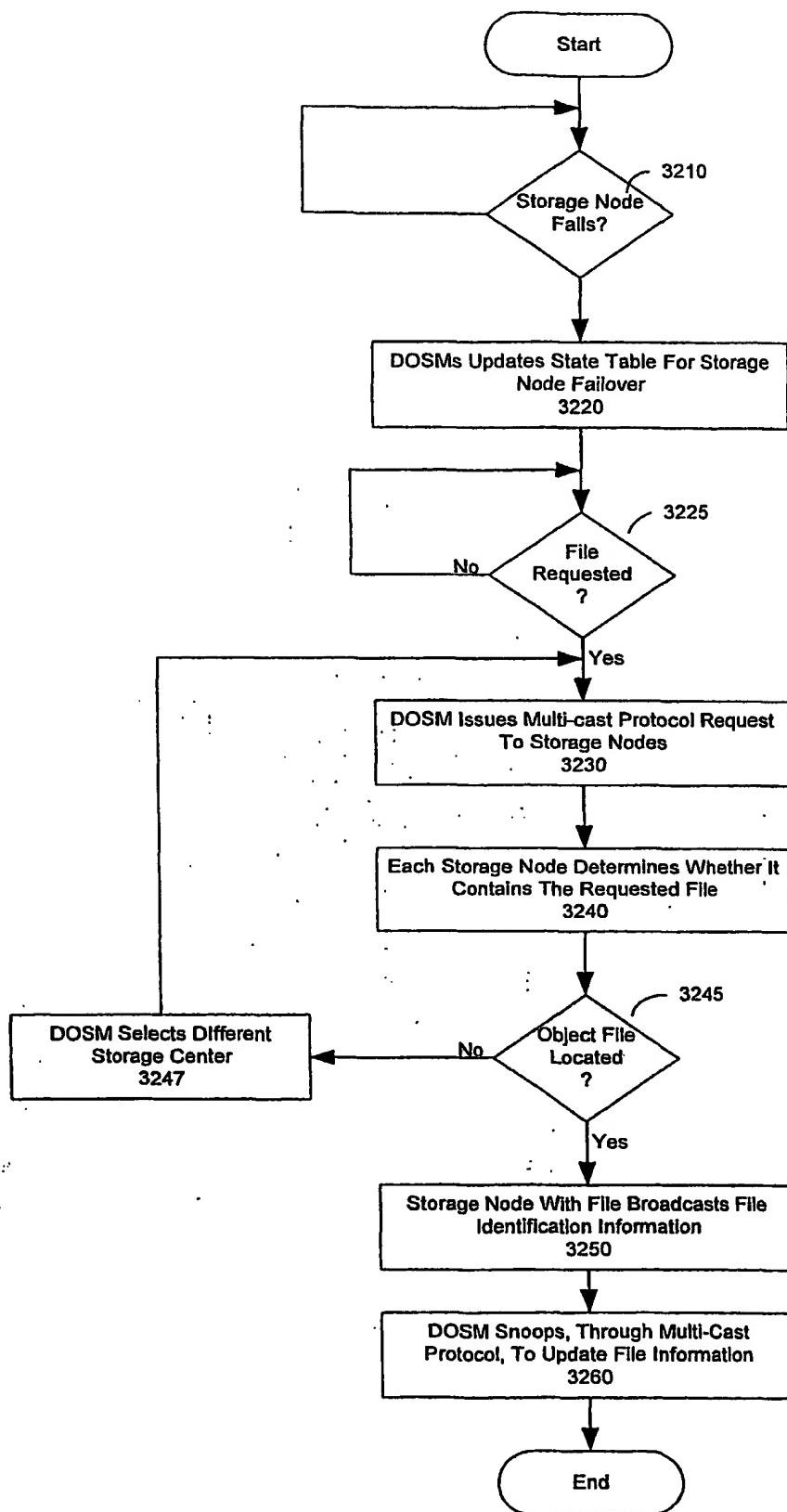


Figure 30

32/32

**Figure 31**

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- BLACK BORDERS**
- IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- FADED TEXT OR DRAWING**
- BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- SKEWED/SLANTED IMAGES**
- COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- GRAY SCALE DOCUMENTS**
- LINES OR MARKS ON ORIGINAL DOCUMENT**
- REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.